

基于实际生产环境，从Linux虚拟化、集群、服务器故障诊断与排除、系统安全性等多角度阐述构建高可用Linux服务器的最佳实践

资深Linux/Unix系统管理专家兼架构师多年一线工作经验结晶，51CTO和ChinaUnix等知名社区联袂推荐



余洪春 著

Build High Availability Linux Servers

构建高可用 Linux服务器



机械工业出版社
China Machine Press

如何构建高可用的Linux服务器，这是很多Linux系统管理员和运维人员都感兴趣的话题，这也是他们努力学习的方向之一。本书作者在该领域浸淫多年，在大量的一线生产环境中积累了丰富的经验和最佳实践。难能可贵的是，他将这些宝贵的知识系统地梳理后总结在了这本书中，旨在与所有同行分享。如果你是一位Linux系统管理员，或运维工程师，或项目实施工程师，只要你细心研读本书的内容并跟随书中的大量案例去实践，相信一定会受益匪浅。

—— 刘天斯 腾讯系统架构师

在IT领域，好书很多，烂书也不少，毫无疑问，本书是一本好书。建议大家用最强悍的执行力来学习本书中的案例，用最细腻的心思去揣摩每个案例背后的原理。如果你能吃透这本书，你就能在短短几个月的时间内将作者几年来积累的知识和经验化为己有，从而在短时间内使自己的技能发生质的飞跃。

—— 曹亚孟 合力金软运维经理

本书的内容全部来自于企业的实际生产环境，非常注重实践性和实用性，书中的所有案例都可以供大家在解决实际问题时参考和借鉴。本书从Linux服务器的虚拟化、生产环境下服务器的故障诊断与排除、生产环境下的SHELL脚本、高可用Linux集群建设、VPN在企业中的部署应用、Linux防火墙等多个方面阐述了构建高可用Linux服务器的方法与最佳实践。强烈推荐！

—— 崔晓辉 大众网高级系统管理员

本书作者从事Linux运维相关的工作已达7年之久，不仅主导过多个Linux集群相关的项目，而且还从事过Linux教学方面的工作，积累了相当丰富的经验。利用工作之余，他把自己多年来积累的实践经验整理到了这本书中，可谓是精华中的精华！本书结合来自一线生产环境的真实案例讲解了Linux集群、Xen虚拟化、iptables的企业级应用和系统安全相关的内容，同时还包括一些常见问题和故障的排除方法。本书尤其适合那些已经有2~3年Linux服务器管理与运维经验的读者，相信本书能在工作中助大家一臂之力。

—— 侯心刚 巨人网络运维中心运维部经理

客服热线: (010) 88378991, 88361066
购书热线: (010) 68326294, 88379649, 68995259
投稿热线: (010) 88379604
读者信箱: hzsj@hzbook.com

华章网站 <http://www.hzbook.com>

 网上购书: www.china-pub.com



定价: 79.00元



Build High Availability Linux Servers

构建高可用 Linux服务器

余洪春 著



机械工业出版社
China Machine Press

资深 Linux/Unix 系统管理专家兼架构师多年一线工作经验结晶, 51CTO 和 ChinaUnix 等知名社区联袂推荐。结合实际生产环境, 从 Linux 虚拟化、集群、服务器故障诊断与排除、系统安全性等多角度阐述构建高可用 Linux 服务器的最佳实践。本书实践性非常强, 包含大量企业级的应用案例及相应的解决方案, 读者可以直接用这些方案解决在实际工作中遇到的问题。

全书一共 10 章。第 1 章以作者的项目实践为基础, 以 RHEL 和 Centos 为平台, 有针对性地讲解了构建高性能 Linux 服务器的应该掌握的核心知识, 包括硬件、网络配置、日志管理、性能优化、监控等重要内容; 第 2 章十分详尽地讲解了 FreeBSD8.1 在企业中的部署与应用, 这是目前第一手关于 FreeBSD8.1 的宝贵资料; 第 3 章讲解了 Linux 服务器的虚拟化, 主要包括 VMware 和 XEN 两大虚拟机在 Windows Server 2003 和 Centos 系统下的使用方法和工作原理, 同时还介绍了 Citrix XenServer 的使用方法; 第 4 章探讨了生产环境下各种棘手的服务器故障的诊断与排除方法; 第 5 章介绍了生产环境下的 SHELL 脚本, 这些脚本都经过实践验证, 读者可以直接在实际工作中使用; 第 6 章首先讲解了构建高可用 Linux 集群的理论知识, 然后以作者的实际项目为例详细演示了构建高可用 Linux 集群环境的方法 (附有项目施工图); 最后还探讨了 MySQL 数据库性能优化方面的话题; 第 7 章以理论与案例相结合的方式讲解了 VPN 在企业中的部署与应用, 包括 VPN 技术的分类和选择、IPSec VPN 的不足和 OpenVPN 的应用范畴、OpenVPN 的部署案例和部署时的注意事项; 第 8 章全面讲解了 Linux 防火墙及系统安全方面的内容, 其中 iptables 相关的知识是重点, 讲解非常详细, 很多脚本都可以直接使用; 第 9 章介绍了构建免费开源的企业级邮件系统的完整过程, 这也来自于作者在实际工作中的实践; 第 10 章针对系统管理员的学习、工作以及职业规划给出了一些宝贵的建议, 对新人尤为有帮助。

封底无防伪标均为盗版

版权所有, 侵权必究

本书法律顾问 北京市展达律师事务所

图书在版编目 (CIP) 数据

构建高可用 Linux 服务器 / 余洪春著. —北京: 机械工业出版社, 2011. 10

ISBN 978-7-111-36055-1

I. 构… II. 余… III. UNIX 操作系统—网络服务器 IV. TP316.81

中国版本图书馆 CIP 数据核字 (2011) 第 202386 号

机械工业出版社 (北京市西城区百万庄大街 22 号 邮政编码 100037)

责任编辑: 杨绣国 陈佳媛

北京京师印务有限公司印刷

2012 年 1 月第 1 版第 2 次印刷

186mm × 240mm · 37.25 印张

标准书号: ISBN 978-7-111-36055-1

定价: 79.00 元

凡购本书, 如有缺页、倒页、脱页, 由本社发行部调换

客服热线: (010) 88378991; 88361066

购书热线: (010) 68326294; 88379649; 68995259

投稿热线: (010) 88379604

读者信箱: hzjsj@hzbook.com



推荐序一

我与“抚琴煮酒”的缘分是从他的一篇博客文章^①开始的。

同大多数做运维的兄弟一样，我平时的工作比较忙，所以很少去网上看技术文章。此外，网上的文章鱼龙混杂，好的技术文章实在难得一见。突然有一天，我的一位学生将“我们的网站压力究竟在哪里？”这篇文章的地址发给了我，并告诉我这篇文章写得很好。于是我忙里偷闲打开看了一下，而且连续看了两遍。在这篇博文中，作者没有使用华丽的辞藻，也没有讲那些不切实际的高深得让读者云里雾里的技术，而是在用最简单朴实的能让每个读者都能懂的语言描绘一个基于生产环境的千万级规模的网站系统的架构情况。看完这篇文章后，我的第一感觉就是又将认识一位真正做运维的同行和知己。紧接着，我把这篇文章转给了我的几百个学生，并开始写评论，结果一不小心写得太长，无法当做评论发表，最后只好贺了一篇博文“如何才能做到网站高并发访问？”^②就这样，我和“抚琴煮酒”相识了。

我和“抚琴煮酒”主要通过博客和 QQ 沟通。虽然我们相识只有短短的几个月，但我们聊天的信息量几乎比普通朋友 3 到 5 年的信息量还要大。在与作者交流的过程中，我发现作者对运维技术有着特殊的执着和热爱，他的心血之作在字里行间透露着自己多年来在点滴的工作中汇集的经验和智慧。有时候，我会对书中的部分内容有些疑问，即便是午夜，他也会给我这样的答复：“做技术要认真，我现在就测试，一会儿给你答复”。

“养儿方知父母恩”，同样，只有写过书的人才知道写书的不易。运维工程师是一个很特殊的职业，它要求我们必须有比程序员和 DBA 更广的知识面，比如系统、开发、网络、数据库等方面的知识都要求掌握。也因为此，国内运维方面的优秀专著屈指可数。在我多年的系统运维培训生涯中，我也仅仅只把鸟哥的书介绍给了我的学生。不是因为鸟哥的书写得有多么高深，辞藻有多么华丽，而是因为鸟哥的书确实更通俗易懂，不但把简单的技术写得有血有肉，而且做到了使复杂的技术深入浅出，让初学者能够迅速掌握。

很幸运的是，“抚琴煮酒”的书也具有这方面的特质。更难得的是，他的书与时俱进，不但讲

① 文章名称：“我们的网站压力究竟在哪里？”链接地址：<http://andrewyu.blog.51cto.com/1604432/612032>。

② 链接地址：<http://oldboy.blog.51cto.com/2561410/615721>。

了鸟哥的书中没有讲的大量基础内容，而且还以实战案例为载体将自己近几年在各企业中常用的负载均衡高可用架构图文并茂地呈现给了广大的读者，并对相关的技术进行了合理的延伸和扩展，比如 keepalive + drbd + nfs 存储的高可用架构。

纵观全书，它就是若干相对很完整的千万级 PV 规模网站的系统架构解决方案的集锦，这些都是作者多年工作经验的结晶。对于战斗在运维一线的兄弟们来说，本书的出版可谓是他们之福。由于运维技术的特殊性，以及写作时间和作者经历等方面的原因，这本书也不可能做到面面俱到，更不可能做到没有瑕疵，只希望大家从此书中找到自己需要的内容，这些内容不仅仅只有技术，还有作者多年如一日的勤奋努力和对技术的热情与执着，写作本书的过程中对技术细节认真负责的态度，以及乐于助人的分享精神等，这些也许比技术本身更宝贵，更重要。

看一本好书，交一个知己，一生足矣。

老男孩 老男孩 Linux 实战运维培训中心总裁

2011 年 9 月



推荐序二

每天的运维工作中都会遇到各种各样的系统问题，比如：

我们如何才能利用已经掌握的 SHELL 相关的知识更好地、自动地完成任务呢？

对于公司而言，我们如何能保证在部分服务器出现故障的情况下业务不会被中断呢？

面对服务器资源有限的现实情况，我们又该如何充分利用有限资源来满足业务的需求呢？

对于服务器安全，我们又能采取哪些措施呢？

.....

有经验的朋友可能都知道，这些问题都是在日常的运维工作中会经常遇到的。这些问题该如何解决？你是不是也经常被这些问题困扰？本书会给你想要的答案。本书针对诸如此类的问题进行了详细的探讨，不是理论的，也不是说教的，而是建立在余洪春先生 7 年的实践经验上的，都是经过实际生产环境所验证的。

本书是余洪春先生在系统运维领域多年工作、实践和探索的结晶。他根据自己的经验，由浅入深地讲述了运维中所面临的种种困难和挑战，从基础的 SHELL 脚本、故障处理，到企业级 Linux 集群应用、虚拟化技术，以及系统安全方面的配置和规划等，面面俱到。如果你所在的生产环境存在的问题与本书中描述的问题类似，便可以直接用书中的解决方案去解决你的实际问题，这些解决方案都是在实际生产环境中被使用和验证过的。所以，无论你是初级的系统管理员，还是资深的运维工程师，在仔细研读本书后，都会有所收获的。

刘晗昭 昆仑万维高级架构师

2011 年 8 月



推荐序三

面对迅猛增长的访问量，作为系统运维人员，我们如何才能保证网站的高并发、高可用、高性能、可扩展性和安全性？这些话题应该是每一位运维人员都关注的，余洪春先生的这本书结合实际的生产环境详细地探讨了这些方面的话题。

经常在 51CTO 上看到余洪春先生发表相关的文章，文章的内容给我的最大感觉就是实用，基本上都是从一线生产环境中总结出来的经验，实践性和可读性都非常强。尤为值得一提的是，在 51CTO 举办的“世博 IT 魔方”活动中，余洪春先生设计了一个用“Nginx + Keepalived”实现的在线票务系统，当时得到了大家的一致好评，此后，关注他的人越来越多。

因为工作的需要，我一直在研究 Web 集群，我和余洪春先生在 51CTO 上成为了好友，并且经常在线上与他讨论一些集群方面的问题，LVS + Keepalived、Nginx + Keepalived、HAProxy 等高可用的解决方案都是我们讨论的焦点。他在多年的项目实施经历中积累了丰富的经验，他与朋友合作运营的网站“一拍网”便是基于 LVS + Keepalived + Nginx + Tomcat7 架构的，目前该网站正稳定运行中。该网站的相关配置也收录到了书中，可以供那些关注 Web 集群的从业人员、爱好者们借鉴，从而让自己少走弯路。

在此，我极力推荐本书，书中包含大量来自一线生产环境的实际案例，毫无保留地与大家分享了如何构建高可用的 Linux 服务器的经验和心得，实在是难得。希望有更多的同道中人和即将从事系统运维和管理的朋友都能从本书中受益。

胡安伟 金游数码运维主管

2011 年 8 月



前言

我的系统管理员之路

2002 年我初识 Linux，那会儿刚毕业，在一家大型国营公司值守 Windows Server 2000 服务器，当时“震荡波”和“冲击波”这两种病毒很是猖狂，没有打补丁的机器无一幸免。我所值守的服务器也未能例外，虽然我们在防毒方面投入了大量的精力和金钱（当时购买的都是正版 Windows 2000 系统和正版瑞星杀毒软件）。有一次去朋友公司（省太平洋寿险下面的一个分支机构）参观，我发现他们的服务器和终端系统都很奇怪，一问才知道是 BSD 系统，因为运行机制不一样，所以 Windows 下的病毒丝毫影响不了它们，而且这些服务器很稳定，基本上不宕机。当时很是羡慕，心想要是哪一天我们的服务器也要换成 BSD 系统，这样“冲击波”和“震荡波”就奈何不了我们，而且也不会蓝屏，那该多好啊。

后来有幸到北京一家大型广告公司上班，公司所用服务器基本上都是 CentOS 和 FreeBSD，内部用的文件服务器是 Samba，Web 服务器是 Apache 和 Nginx，公司的 NAT 路由器是 iptables，核心业务是 CDN 系统，几乎全部装的都是 CentOS 5.1 x86_64，仅有一台装的是 Windows Server 2003，供程序员开发 .NET 程序之用。公司的这套 CDN 系统要负责处理所有的流量，即使在高峰期并发量特别大的时候，网站也非常稳定。

我当时也对 Linux 产生了浓厚的兴趣，尝试改掉自己多年使用 Windows 的习惯，换成了纯字符操作，并且尝试用 SHELL 完成自动化工作，用 vim + sed 处理文档。后来，我发现自己越来越喜欢 Linux 了。公司有一台 vsftpd 服务器，3 年没有重启了，这很让我吃惊，所以多次与人聊起。后来我又得知一位朋友所在的公司有一台很老的 RH8 服务器，因为负责的是公司的核心业务，已经 9 年没有重启了，当时更加感到震撼，这更加坚定了我要学好 Linux 的决心。

在做项目实施工程师期间，有些客户的核心网站的并发量并不是太大，但比较重要，所以他们都要求部署 Linux 集群，有时指定要部署 LVS 或 HAProxy。在项目实施的过程中，我发现 LVS/HAProxy 的负载均衡确实非常强大，可以与硬件级的 F5 负载均衡器的能力相媲美。很快我就被 Linux 集群这门艺术迷住了，我自己曾研究了 Nginx + Keepalived 这种负载均衡高可用架构，并且在许多项目中成功实施，客户反映效果也不错。

现阶段我的工作主要是维护“一拍网”和另外一个大型电子商务网站，相对于 CDN 系统而言，它没有节点冗余，所以对 Linux 集群技术的要求更高。我负责的网站基本都做了双机高 HA，LVS + Keepalived 和 Nginx + Keepalived，还有 DRBD + Heartbeat 及 MySQL 的 replication 复制。另外，由于电子商务网站涉及钱的问题，所以对安全性的要求也很高，我们平时都会从网络安全（硬件防火墙）、系统安全、代码安全和数据库安全这些方面着手，尽力避免一切影响网站安全的行为。工作虽然辛苦，但看着自己架构的网站顺利稳定地运行，心里还是很有成就感的，这也是我的工作动力的主要源泉。

写作本书的目的

从事系统管理和系统集成方面的工作已有七八年了，在工作期间，我曾经担任了一段时间的红帽 RHCE 讲师，到东北大学等高校推广红帽 Linux 及 FreeBSD 等开源系统。在教学过程中我发现，很多学生在进入企业后都无法胜任自己的工作，更谈不上正确地规划自己的职业道路了。一方面因为企业的生产环境具有一定的复杂性；另一方面市场上入门书居多，缺乏能真正指导读者解决实际问题的书籍。例如，很多书都只是通过 VMware ESXI 或 XenServer 虚拟出了 VM 系统，对于线上环境，根本没有涉及并发、PV 和数据库压力等相关话题。

之所以写这本书，一方面是想对自己这些年的工作经验和心得进行一次系统的梳理和总结；另一方面是想将自己的经验分享给大家，希望能帮助大家少走弯路。通过本书中的项目实践（包括 Linux 集群、OpenVPN、邮件系统和 iptables 防火墙）和线上环境的 SHELL 脚本，大家能迅速进入工作状态。书中专门用了一个章节来向大家介绍目前应用得比较多的 FreeBSD8.1 x86_64 系统（市场上目前无相关书籍介绍）。书中所提供的 SHELL 脚本和 iptables 脚本均来自于线上的生产服务器，大家均可以直接拿来用。关于 Linux 集群的项目实践，大家也可以根据实际项目的需求直接采用，以此来设计自己公司的网站架构。

希望大家能通过本书掌握 Linux 的精髓，轻松而愉快地工作，从而提高自己的技术水平，这是我非常希望看到的，这也是我写此书的初衷。

读者对象

本书适合以下几类读者：

- ☐ 系统管理员和系统工程师
- ☐ 网络管理员和企业网管
- ☐ 项目实施工程师
- ☐ 开发人员
- ☐ 计算机相关专业的学生

如何阅读本书

本书的内容是对实际工作经验的总结，涉及大量的知识点和专业术语，建议经验还不是很丰富的读者先了解第 1 章和第 2 章的内容，这两章比较基础，如果大家在学习过程中根据这两章的讲解进行操作，定会达到事半功倍的效果。

系统管理员和系统工程师们则可以重点关注第 3 章的（Linux 服务器虚拟化）、第 6 章（Linux 集群）及第 8 章（iptables 防火墙及系统安全）的内容，这些都与我们的日常工作息息相关，建议

大家多花些精力和时间，抱着一切从线上环境去考虑的态度去学习。

对于网络管理员和企业网管来说，如果基础不是太扎实，建议先学习第1章的内容，然后将重点放在第7章（VPN在企业中的部署应用）和第8章上。

对于项目实施工程师而言，由于大多数都是从事系统集成相关工作的，因此建议顺序学习全书的内容，重心可以放在本书的第6章上面。

对于开发人员来说，由于其只需对系统有一个大概的了解，重点可以放在第1章、第2章和第5章（生产环境下的SHELL脚本）上。

大家可以根据自己的职业发展和工作需要选择不同的阅读顺序和侧重点，同时也可以对其他相关的知识点有一定的了解。

致谢

感谢我的家人，她们在生活上对我无微不至的照顾，让我更有精力和动力去工作和创作。

感谢东北大学信息技术学院的付冲教授，感谢您在我人生最穷困潦倒的时候伸出援手。

感谢北京总公司的技术总监唐老师，他对iptables防火墙相关的内容提出了许多指导性的建议，并且在CDN系统维护方面教会了我许多。

感谢老男孩前辈在网站架构设计方面给出的指导性意见，您的经验和专业知识让我受益匪浅。

感谢和我一起从事系统运维工作的朋友：曹亚孟、胡安伟和崔晓辉。曹亚孟为本书中与Linux虚拟化相关的内容提供了宝贵的资料，这是他几年来从事XEN虚拟化工作的经验总结；胡安伟为本书提供了许多精美的插图，并就Linux集群相关的内容提出了许多宝贵的意见；崔晓辉为本书提供了大量的线上SHELL脚本。另外，还要对我的同事Ritto表示衷心的感谢！

感谢朋友刘鑫，和我一起花了大量时间研究和调试HAproxy + keepalived。

感谢51cto.com的编辑们，尤其是赵克衡、杨赛、王文文、张浩和邵程程，正是有了你们的信任和帮助，此书才得以问世。

感谢Linux高级群中的深夜的蚊子、结冰的西瓜和无语、jack、狼希耐、REAL97等朋友，感谢你们对我的支持，没有你们的相伴，我能否坚持到今天，可能还是个未知数。

感谢我的朋友三宝，感谢他这么多年来对我的信任和支持，在我苦闷的时候陪我聊天。

感谢在工作和生活中给予过我帮助的所有人，感谢你们，正是因为有了你们，才有了本书的问世。

关于勘误

尽管我花了大量时间和精力去核对书中的文字、代码和图片，但因为时间仓促和水平有限，书中仍难免会有一些错误和纰漏，如果大家发现问题，恳请反馈给我，相关信息可发到我的邮箱yuhongchun027@gmail.com。尽管我无法保证每一个问题都会有正确的答案，但我肯定会努力回答并指出一个正确的方向。

如果大家对本书有任何疑问或想与我探讨Linux技术，可以访问我的个人博客，地址为：<http://andrewyu.blog.51cto.com>，我的微博地址为：<http://weibo.com/yuhongchun027>。另外，我在51cto.com和ChinaUnix社区的ID均为“抚琴煮酒”，大家也可以直接通过此ID在社区中与我在在线交流。

余洪春（抚琴煮酒）

2011年8月



目 录

推荐序一

推荐序二

推荐序三

前 言

第 1 章 Linux 服务器构建基础 / 1

1.1 Linux 服务器的安装方法 / 2

1.1.1 光盘安装 Centos5.5 x86_64 / 2

1.1.2 使用 PXE + DHCP + Apache + KickStart 无人值守安装 RHEL / 12

1.1.3 Linux 的其他安装方法 / 17

1.2 全面了解 Linux 服务器 / 18

1.2.1 查看 Linux 服务器的 CPU 详细情况 / 18

1.2.2 查看 Linux 服务器的内存使用情况 / 19

1.2.3 查看 Linux 服务器的硬盘使用情况 / 20

1.2.4 查看 Linux 系统的平均负载 / 24

1.2.5 查看 Linux 系统的其他参数 / 25

1.3 Linux 服务器的网络配置 / 28

1.3.1 配置 Linux 服务器的网络 / 28

1.3.2 查看 Linux 服务器的网络连接 / 31

1.3.3 查看 Linux 服务器的进程 / 39

1.3.4 在 Centos5.5、FreeBSD8.1 及 Windows 下添加静态路由 / 43

1.4 Linux 服务器的日志管理 / 45

- 1.4.1 系统日志 syslog.conf 的配置详解 / 46
- 1.4.2 Linux 下的日志维护技巧 / 47
- 1.4.3 用 shell 脚本分析 Nginx 日志 / 51
- 1.5 Linux 服务器的优化 / 53
 - 1.5.1 根据服务器应用来选购服务器 / 54
 - 1.5.2 Centos5.5 最小化安装后的优化 / 58
 - 1.5.3 优化 Linux 下的内核 TCP 参数以提高系统性能 / 63
 - 1.5.4 生产服务器应尽量选择编译安装软件包 / 65
- 1.6 用开源工具 Nagios 监控 Linux 服务器 / 66
 - 1.6.1 Centos5.5 下的监控工具简介 / 66
 - 1.6.2 Nagios 应该监控的服务器基础选项 / 67
 - 1.6.3 Nagios 监控 Windows 2003 时应注意的事项 / 67
 - 1.6.4 用 Nagios 监控 Nginx 脚本 / 68
 - 1.6.5 Nagios 使用心得 / 74
- 1.7 项目实施中应该注意的事项 / 75
- 1.8 小结 / 77

第2章 FreeBSD8.1 在企业中的部署应用 / 78

- 2.1 最小化安装 FreeBSD8.1 / 79
- 2.2 最小化安装 FreeBSD8.1 后的升级优化部署 / 90
 - 2.2.1 最小化安装 FreeBSD8.1 服务器后建议做的事 / 90
 - 2.2.2 系统管理员应该知道的 FreeBSD8.1 的一些事项 / 95
 - 2.2.3 在 FreeBSD8.1 下高效地安装和卸载软件 / 99
 - 2.2.4 查看 FreeBSD8.1 的硬件配置 / 100
- 2.3 在 FreeBSD8.1 下部署 jail 虚拟机 / 103
 - 2.3.1 FreeBSD8.1 下的 jail 概述 / 103
 - 2.3.2 FreeBSD8.1 下安装 jail 的详细步骤 / 104
 - 2.3.3 FreeBSD8.1 下 jail 的管理 / 105
 - 2.3.4 通过 ezjail 来创建和管理 jail 虚拟机 / 106
 - 2.3.5 jail 在生产环境下的注意事项 / 109
- 2.4 在 FreeBSD8.1 下搭建版本控制服务器 / 109
 - 2.4.1 版本控制软件的概念 / 109
 - 2.4.2 在 FreeBSD8.1 下搭建 CVS 服务器 / 109
 - 2.4.3 在 FreeBSD8.1 下搭建 SVN 服务器 / 113
 - 2.4.4 在 FreeBSD8.1 下搭建 Git 服务器 / 117
- 2.5 在 FreeBSD8.1 下搭建 Samba 文件服务器 / 121
 - 2.5.1 Samba 概述 / 121

- 2.5.2 在 FreeBSD8.1 下安装配置 Samba3.4 / 121
- 2.5.3 Samba 的详细语法配置 / 122
- 2.5.4 Samba 在工作中的总结 / 124
- 2.5.5 Linux 下的高级权限文件控制 / 125
- 2.5.6 Samba 在企业开发环境中的常用案例之一 / 127
- 2.5.7 Samba 在企业开发环境中的应用案例之二 / 128
- 2.6 在 FreeBSD8.1 下配置 NFS 文件服务器 / 131
- 2.7 在 FreeBSD8.1 与 Centos5.5 下搭建 rsync 服务器 / 134
 - 2.7.1 rsync 的概念 / 134
 - 2.7.2 在 Centos5.5 下配置 rsync 服务器 / 134
 - 2.7.3 在 FreeBSD8.1 下配置 rsync 服务器 / 138
 - 2.7.4 rsync + Inotify 实现数据的实时同步更新 / 140
- 2.8 在 FreeBSD8.1 下搭建 vsftpd 服务器 / 143
 - 2.8.1 vsftpd 服务器的特点 / 143
 - 2.8.2 vsftpd 的运行模式 / 144
 - 2.8.3 vsftpd 的数据连接模式 / 144
 - 2.8.4 vsftpd 到底安全在哪里 / 145
 - 2.8.5 在 FreeBSD8.1 下配置 vsftpd 服务器 / 146
 - 2.8.6 用 vsftpd 作 Linux/Unix 之间的异地备份 / 147
- 2.9 在 FreeBSD8.1 和 Centos5.5 下搭建 PHP 与 Java 应用环境 / 149
 - 2.9.1 在 FreeBSD8.1 下搭建 FAMP 环境 / 149
 - 2.9.2 在生产环境下配置 LNMP 环境 / 152
 - 2.9.3 在 Centos5.5 下搭建 Java 运行环境 / 172
- 2.10 小结 / 176

第3章 Linux 服务器虚拟化 / 177

- 3.1 在 Windows Server 2003 下安装 VMware GSX Server / 178
- 3.2 用 Windows 2003 + VMware Server 搭建 64 位系统测试环境 / 181
- 3.3 在 Centos5.6 x86_64 下安装 XEN 虚拟机 / 183
 - 3.3.1 XEN 在 Centos5.6 x86_64 下的安装步骤 / 183
 - 3.3.2 XEN 虚拟机的优势 / 185
- 3.4 XEN 在生产环境下的应用 / 185
 - 3.4.1 XEN 虚拟化的基本概念 / 185
 - 3.4.2 在 Centos5.5 下安装 XEN 虚拟机 / 187
 - 3.4.3 安装第一台虚拟机 (模板机) / 189
 - 3.4.4 XEN 寄宿服务器的管理 / 191
 - 3.4.5 XEN 在生产环境下的应用 / 194

3.5 Citrix XenServer5.6 虚拟机试用手记 / 196

3.6 小结 / 202

第4章 生产环境下服务器的故障诊断与排除 / 203

4.1 快速排障的重要性和必要性 / 204

4.2 安装系统时容易发生的错误描述与处理方法 / 204

4.2.1 忘记了 Centos5.5 的 root 密码怎么办 / 204

4.2.2 正确重设 root 密码 / 206

4.2.3 安装 FreeBSD8.1 时不要设置/boot 分区 / 207

4.2.4 Centos5.5 的 Grub 引导程序出错 / 207

4.2.5 安装 Centos5.5 时忘了关闭 iptables 和 SELinux / 208

4.2.6 如何解决 Putty 或 PieTTY 的乱码问题 / 209

4.2.7 安装双系统时不小心删除了 Grub 所在的分区 / 209

4.3 网络配置时容易发生的错误描述与处理方法 / 211

4.3.1 安装 Centos5.5 时忘了激活网卡 / 211

4.3.2 Centos5.5 网卡文件备份的正确方法 / 212

4.3.3 解决远程桌面超出最大连接数的问题 / 213

4.3.4 在 Centos5.5 下如何正确配置网关 / 214

4.3.5 VMware 的机器应该如何配置自动对时 / 214

4.3.6 防火墙初始化的注意事项 / 215

4.4 系统维护时的注意事项 / 215

4.4.1 尽量源码安装, 谨慎操作 yum / 215

4.4.2 服务器硬件改动进入了 Emergency 模式 / 216

4.4.3 如何以普通用户的身份编辑无权限的文件 / 216

4.4.4 在 Linux 下配置最大文件打开数的方法 / 216

4.4.5 在 Crontab 下运行 PHP 程序的正确方法 / 218

4.4.6 在 Crontab 下正确防止脚本运行冲突 / 218

4.5 紧急处理线上服务器故障的办法 / 219

4.5.1 更改 Administrator 密码导致计划任务无法执行 / 219

4.5.2 FreeBSD8.1 下的 sudoer 文件意外损坏 / 219

4.5.3 Centos5.5 的 root 密码被恶意篡改 / 219

4.5.4 bash 损坏该如何正确处理 / 220

4.5.5 正确操作 nohup 让程序始终在后台运行 / 221

4.5.6 负载均衡器出现故障 / 221

4.6 检查机房应注意的位置和细节问题 / 221

4.7 系统维护时应注意的非技术因素 / 222

4.8 小结 / 222

第5章 生产环境下的 SHELL 脚本 / 223

- 5.1 Vim 的基础用法及进阶心得 / 224
- 5.2 Sed 的基础用法及实用举例 / 228
 - 5.2.1 Sed 的基础语法格式 / 228
 - 5.2.2 Sed 的用法举例说明 / 230
- 5.3 基础正则表达式 / 235
- 5.4 Linux 下强大的查找命令 find / 240
- 5.5 汇总 Linux/Unix 下的 bash 快捷键 / 248
- 5.6 生产环境下的 SHELL 脚本分类 / 249
 - 5.6.1 生产环境下的 SHELL 脚本备份类 / 250
 - 5.6.2 生产环境下的开发类 SHELL 脚本 / 257
 - 5.6.3 生产环境下的统计类 SHELL 脚本 / 259
 - 5.6.4 生产环境下的监控类 SHELL 脚本 / 262
 - 5.6.5 生产环境下的自动化类 SHELL 脚本 / 265
 - 5.6.6 生产环境下的安全类 SHELL 脚本 / 269
- 5.7 小结 / 272

第6章 构建高可用的 Linux 集群 / 273

- 6.1 负载均衡高可用的核心概念和常用软件 / 274
 - 6.1.1 什么是负载均衡高可用 / 274
 - 6.1.2 以 F5 BIG-IP 作为负载均衡器 / 275
 - 6.1.3 以 LVS 作为负载均衡器 / 275
 - 6.1.4 以 Nginx 作为负载均衡器 / 281
 - 6.1.5 以 HAProxy 作为负载均衡器 / 281
 - 6.1.6 高可用软件 Keepalived / 283
 - 6.1.7 高可用软件 Heartbeat / 283
 - 6.1.8 高可用块设备 DRBD / 284
- 6.2 负载均衡中的名词解释 / 285
 - 6.2.1 什么是 Session / 285
 - 6.2.2 什么是 Session 共享及实现的方法 / 285
 - 6.2.3 什么是会话保持 / 286
- 6.3 负载均衡器的会话保持机制 / 287
 - 6.3.1 F5 Big-IP 的会话保持机制 / 287
 - 6.3.2 LVS 的会话保持机制 / 288
- 6.4 Linux 集群的项目案例分享 / 299
 - 6.4.1 项目案例一：用 Nginx + Keepalived 实现在线票务系统 / 299

- 6.4.2 项目案例二：企业级 Web 负载均衡高可用之 Nginx + Keepalived / 302
- 6.4.3 项目案例三：用 LVS + Keepalived 构建高可用 JSP 集群 / 313
- 6.4.4 项目案例四：生产环境下的高可用 NFS 文件服务器 / 322
- 6.4.5 项目案例五：HAProxy 双机高可用方案之 HAProxy + Heartbeat / 331
- 6.5 项目实践中 Linux 集群的总结和思考 / 336
- 6.6 网站架构应关注和研究的方向 / 338
- 6.7 MySQL数据库的优化 / 339
 - 6.7.1 服务器物理硬件的优化 / 339
 - 6.7.2 MySQL 应该采用编译安装的方法 / 340
 - 6.7.3 MySQL 配置文件的优化 / 340
 - 6.7.4 MySQL 上线后根据 status 状态进行适当优化 / 346
 - 6.7.5 MySQL 数据库的可扩展性架构方案 / 352
 - 6.7.6 MySQL 数据库的 Replication 高可用架构 / 352
 - 6.7.7 MySQL Cluster 集群配置方案 / 354
 - 6.7.8 生产环境下的 MySQL 数据库主从 Replication 同步 / 360
 - 6.7.9 可扩展性设计之数据切分 / 368
- 6.8 生产环境下的 MySQL 数据库备份 / 369
- 6.9 部分项目施工图纸 / 372
- 6.10 小结 / 374

第7章 VPN 在企业中的部署应用 / 375

- 7.1 流行的 VPN 技术及其分类 / 376
- 7.2 如何选择自己需要的 VPN / 378
- 7.3 IPSec VPN 的不足 / 378
- 7.4 OpenVPN 的应用范畴 / 379
- 7.5 经典企业 VPN 部署案例 / 379
 - 7.5.1 案例一：在 Centos5.5 x86_64 下单网卡配置 PPTPD 服务器 / 379
 - 7.5.2 案例二：在 Centos5.5 x86_64 下路由模式配置 OpenVPN 服务器 / 386
 - 7.5.3 案例三：在 FreeBSD8 x86_64 下网桥模式配置 OpenVPN 服务器 / 396
- 7.6 部署 OpenVPN 服务器的注意事项 / 402
 - 7.6.1 OpenVPN 如何注销用户 / 402
 - 7.6.2 OpenVPN 服务器的安全问题 / 403
 - 7.6.3 OpenVPN 服务器的负载均衡 / 404
- 7.7 小结 / 404

第8章 Linux 防火墙及系统安全 / 405

8.1 基础网络知识 / 406

8.1.1 OSI 网络参考模型 / 406

8.1.2 TCP/IP 三次握手/四次挥手的过程详解 / 407

8.1.3 其他基础网络知识 / 409

8.2 Linux 防火墙的概念 / 409

8.3 Linux 防火墙在企业中的作用 / 410

8.4 Linux 防火墙的语法 / 410

8.5 iptables 的基础知识 / 414

8.5.1 iptables 的状态 state / 414

8.5.2 iptables 的 Conntrack 记录 / 416

8.5.3 关于 iptables 模块的说明 / 416

8.5.4 iptables 防火墙初始化的注意事项 / 416

8.5.5 如何保存运行中的 iptables 规则 / 417

8.6 如何流程化编写 iptables 脚本 / 418

8.7 学习 iptables 应该掌握的工具 / 420

8.7.1 命令行的抓包工具 TCPDump / 420

8.7.2 图形化抓包工具 Wireshark / 421

8.7.3 强大的命令行扫描工具 Nmap / 424

8.7.4 安全工具 hping / 426

8.8 iptables 的简单脚本学习 / 428

8.8.1 普通的 Web 主机防护脚本 / 429

8.8.2 如何让别人 ping 不到自己而自己 ping 通别人 / 430

8.8.3 建立安全的 vsftpd 服务器 / 432

8.9 线上生产服务器的 iptables 脚本 / 436

8.9.1 安全的主机 iptables 防火墙脚本 / 437

8.9.2 自动分析黑名单及白名单的 iptables 脚本 / 439

8.9.3 利用 recent 模块限制同一 IP 的连接数 / 441

8.9.4 利用 DenyHosts 工具和脚本来防止 SSH 暴力破解 / 444

8.9.5 将 iptables 作为企业的 NAT 路由器 / 448

8.9.6 如何使用工具精确地监控 NAT 路由器 / 451

8.10 TCP_wrappers 应用级防火墙的介绍和应用 / 458

8.11 工作中的 Linux 防火墙总结 / 460

8.12 Linux 系统自身的安全防护 / 461

8.12.1 SELinux 简介 / 461

8.12.2 SELinux 的相关设置 / 461

- 8.13 Linux 系统安全相关的工具 / 462
 - 8.13.1 Rootkit 检测工具 Chkrootkit / 462
 - 8.13.2 文件系统完整性检查工具 Tripwire / 464
 - 8.13.3 防恶意扫描软件 PortSentry / 470
- 8.14 Linux 服务器基础防护篇 / 474
- 8.15 如何防止入侵 / 475
- 8.16 小结 / 476

第9章 如何构建开源免费的企业级邮件系统 / 477

- 9.1 DNS 服务器的架设 / 478
 - 9.1.1 邮件服务器与 DNS 的关系 / 478
 - 9.1.2 如何架设内部 DNS 服务器 / 480
 - 9.1.3 如何以源码方式安装公网 DNS 服务器 / 487
 - 9.1.4 维护 DNS 服务器应该注意的事项 / 494
- 9.2 电子邮件的传输过程 / 496
- 9.3 如何搭建开发邮件服务器 / 498
 - 9.3.1 搭建 Sendmail + Dovecot 邮件系统 / 498
 - 9.3.2 搭建 Postfix + Dovecot 邮件系统 / 501
- 9.4 搭建 iRedmail 企业级邮件服务器 / 504
 - 9.4.1 iRedmail 企业级邮件服务器的介绍 / 504
 - 9.4.2 在 Centos5.2 x86_64 上安装 iRedmail0.4.0 / 505
 - 9.4.3 Postfix 本身的防垃圾功能 / 508
 - 9.4.4 iRedmail0.4.0 特有的防垃圾技术 / 513
 - 9.4.5 iRedmail0.4.0 是如何利用 ClamAV 防病毒的 / 517
 - 9.4.6 iRedmail0.4.0 邮件服务器的网络安全 / 517
 - 9.4.7 iRedmail0.4.0 邮件服务器系统的监控 / 520
 - 9.4.8 iRedmail0.4.0 的系统文件备份 / 522
 - 9.4.9 iRedmail0.4.0 的 MySQL 数据库备份方案 / 526
 - 9.4.10 维护 iRedmail0.4.0 邮件服务器的一些注意事项 / 532
- 9.5 小结 / 533

第10章 系统管理员在企业中的职业定位及发展方向 / 534

- 10.1 系统管理员的概念和工作职责 / 535
- 10.2 系统管理员应该熟悉的系统 / 536
- 10.3 系统管理员应该熟悉的工具 / 540
- 10.4 Linux 的学习及进阶之路 / 544

- 10.5 系统管理员应该如何工作 / 547
- 10.6 给 Linux/Unix 新人的建议 / 549
- 10.7 系统管理员之企业生存守则 / 550
- 10.8 小结 / 553

附录 A Xmanager 3.0 企业版实用技巧集锦 / 554

附录 B 使用 Screen 管理远程会话 / 564

附录 C 自动化部署管理工具 Puppet / 566

附录 D 漫谈 CDN 系统运维与电子商务运维 / 572



第 1 章 Linux 服务器构建基础

- 1.1 Linux 服务器的安装方法
- 1.2 全面了解 Linux 服务器
- 1.3 Linux 服务器的网络配置
- 1.4 Linux 服务器的日志管理
- 1.5 Linux 服务器的优化
- 1.6 用开源工具 Nagios 监控 Linux 服务器
- 1.7 项目实施中应该注意的事项
- 1.8 小结

作为一名系统工程师和项目实施工程师，我的工作内容主要有电子商务网站的运维、内网开发环境的部署、公司外包项目的实施等。在这些工作中，我用到的系统绝大多数是免费开源的 Centos 系列，它的稳定和高效让我印象深刻。本章将以 64 位的 Centos5.5 的生产服务器为平台，逐步介绍它的安装方法、网络配置、日志分析、性能及状态监控，以及它的优化及虚拟化等内容。这些都是构建高性能及高可用 Linux 系统的基础，希望对大家有所帮助。另外，Centos5.5 的安装过程和步骤，同样适合其他的 Linux 系统，如 RHEL 系列、SUSE Linux Enterprise、Ubuntu Server。在本章和以后章节中演示所用的所有系统均是 64 位的，如 64 位的 Centos5.5（对于它有两种表示方法，一种是 64bit Centos5.5；另外一种也为大家所接受，即 Centos5.5 x86_64。其他 Linux 系统的表示法与此类似）。

1.1 Linux 服务器的安装方法

Centos5.5 x86_64 的安装方法很多，最常见的有光盘安装、无人值守安装以及 U 盘安装，还有 ISO 硬盘安装等。我们在实际工作中通常以光盘安装和无人值守安装为主，所以这里主要介绍这两种安装方法。

1.1.1 光盘安装 Centos5.5 x86_64

先说说准备工作。为了截图方便，我在笔记本上安装了 VMware Console Client，同时在家用服务器上安装了 Windows 2003 R2 x86_64 + VMware Server1.01。VMware Server1.01 很稳定，所以我将它用在了生产环境下的客户演示系统中，对此感兴趣的朋友也可以关注一下。我将虚拟机的硬盘设定为 10GB，分区情况如下：

```
/ 2048M
/boot 128M
swap 512M
/data 7552M (即剩余的所有硬盘空间)
```

如果是线上的服务器，假设它是 2TB 的 SATA 硬盘、8GB 内存，那么建议按如下方式进行分区：

```
/ 20480M
/boot 128M
swap 10240M
/data 2016152M (即剩余的所有硬盘空间)
```

为了便于演示，我将 VMware Server 中的 Centos5.5 x86_64 虚拟机的硬件按图 1-1 所示的情况进行了分配。

另外，我在笔记本上安装了 VMware Server Console，要连接服务器的 VMware Server 也很简单。运行此程序后输入 Windows 2003 所在机器的 IP 地址和 Administrator 的相应密码即可进入，这样就会像操作本地的 VMware Server 一样方便了，如图 1-2 和图 1-3 所示。

启动了 VMware Server 后，即进入了 Centos5.5 的安装界面，如图 1-4 所示。

接下来会让你选择是否检测光盘媒体是否存在着问题，Skip 表示选择略过。我们在机房安装系统时，常会选择 Skip 以略过检查过程，节约安装时间，如图 1-5 所示。

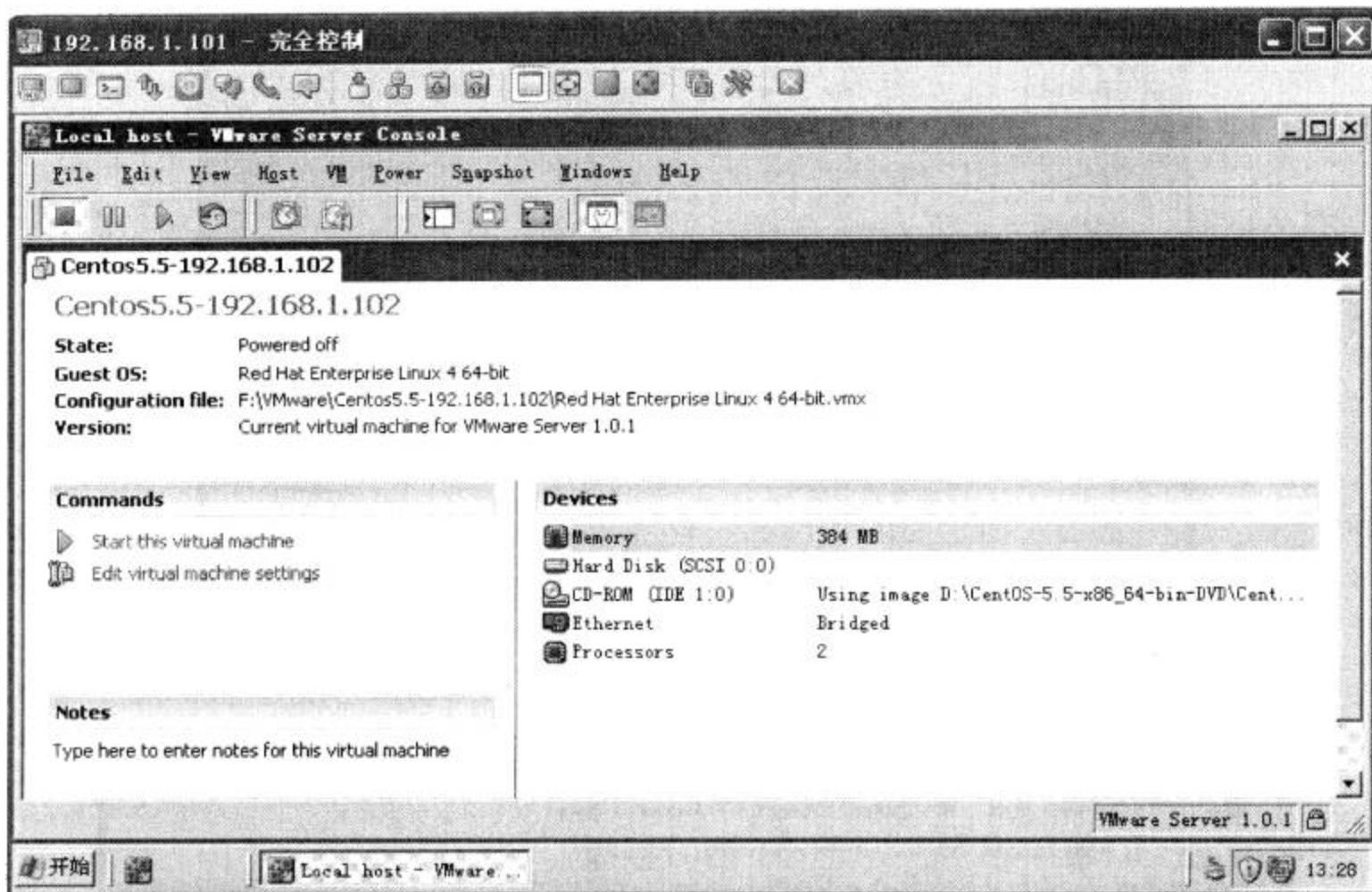


图 1-1 Centos5.5 虚拟机的硬件分配

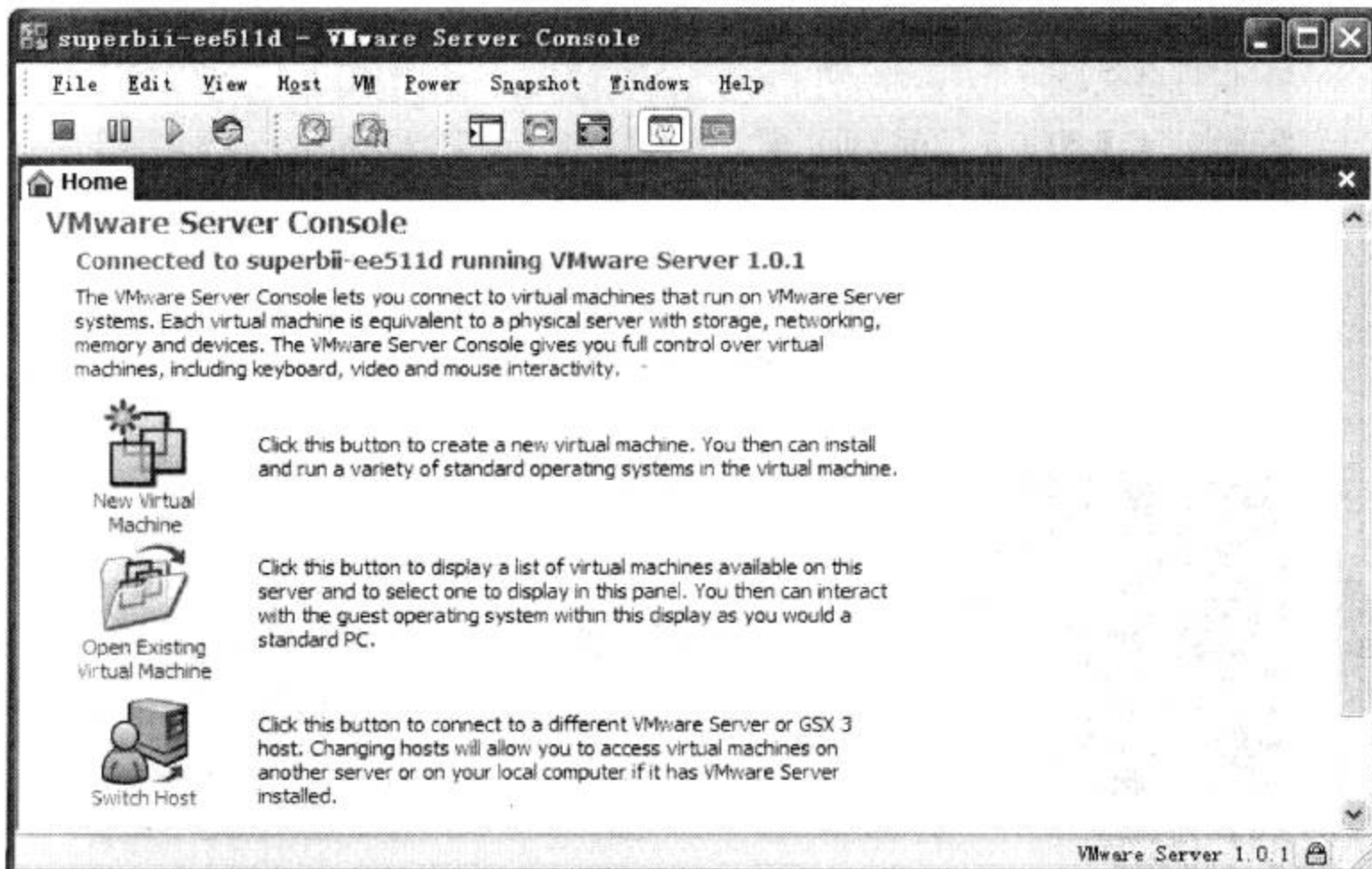


图 1-2 安装 VMware Server Console



图 1-3 利用 VMware Server Console 进入 VMware Server

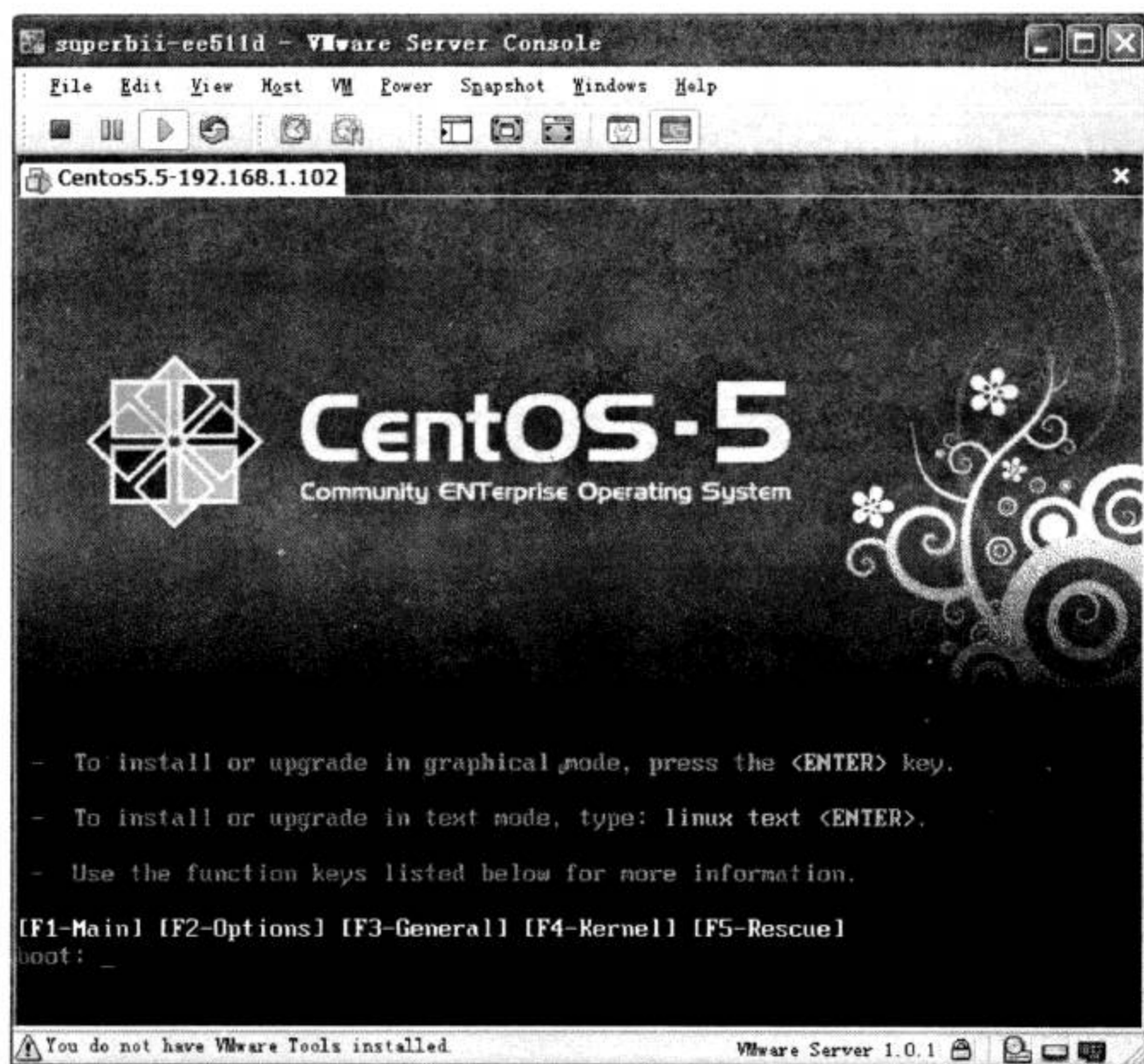


图 1-4 Centos5.5 的安装界面

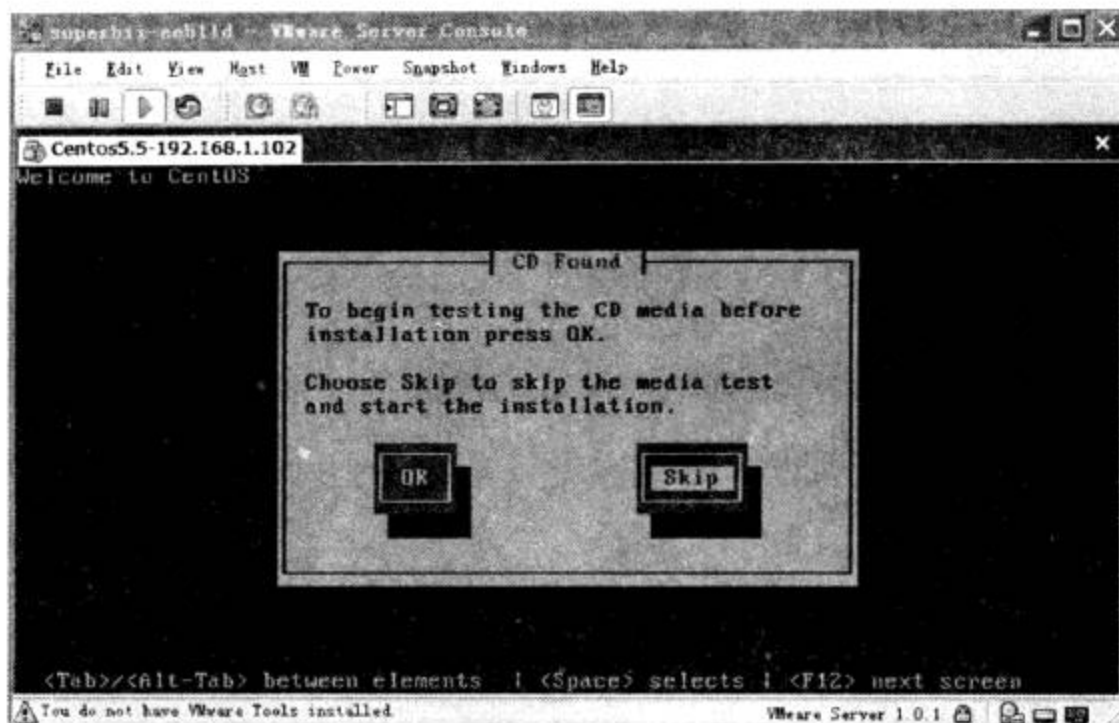


图 1-5 系统提示是否检测光盘媒体

接下来让你选择系统的语言，比如是简体中文、繁体中文，还是英文或其他语言，如图 1-6 所示。根据线上工作的经验，英文是最稳定的。

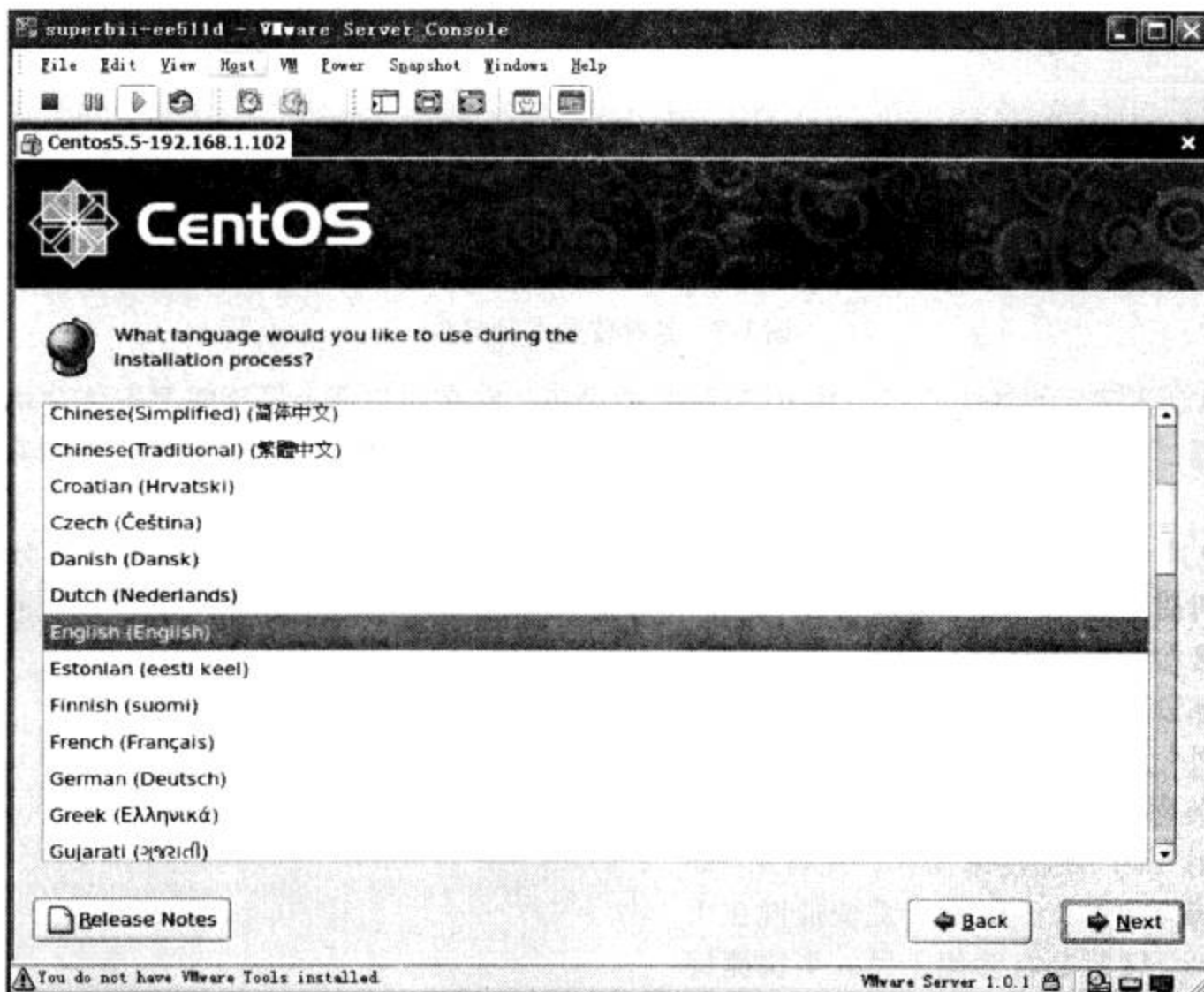


图 1-6 选择系统安装的语言

图 1-7 显示的是选择安装哪种键盘，一般选择 U.S.English（即美式的）。

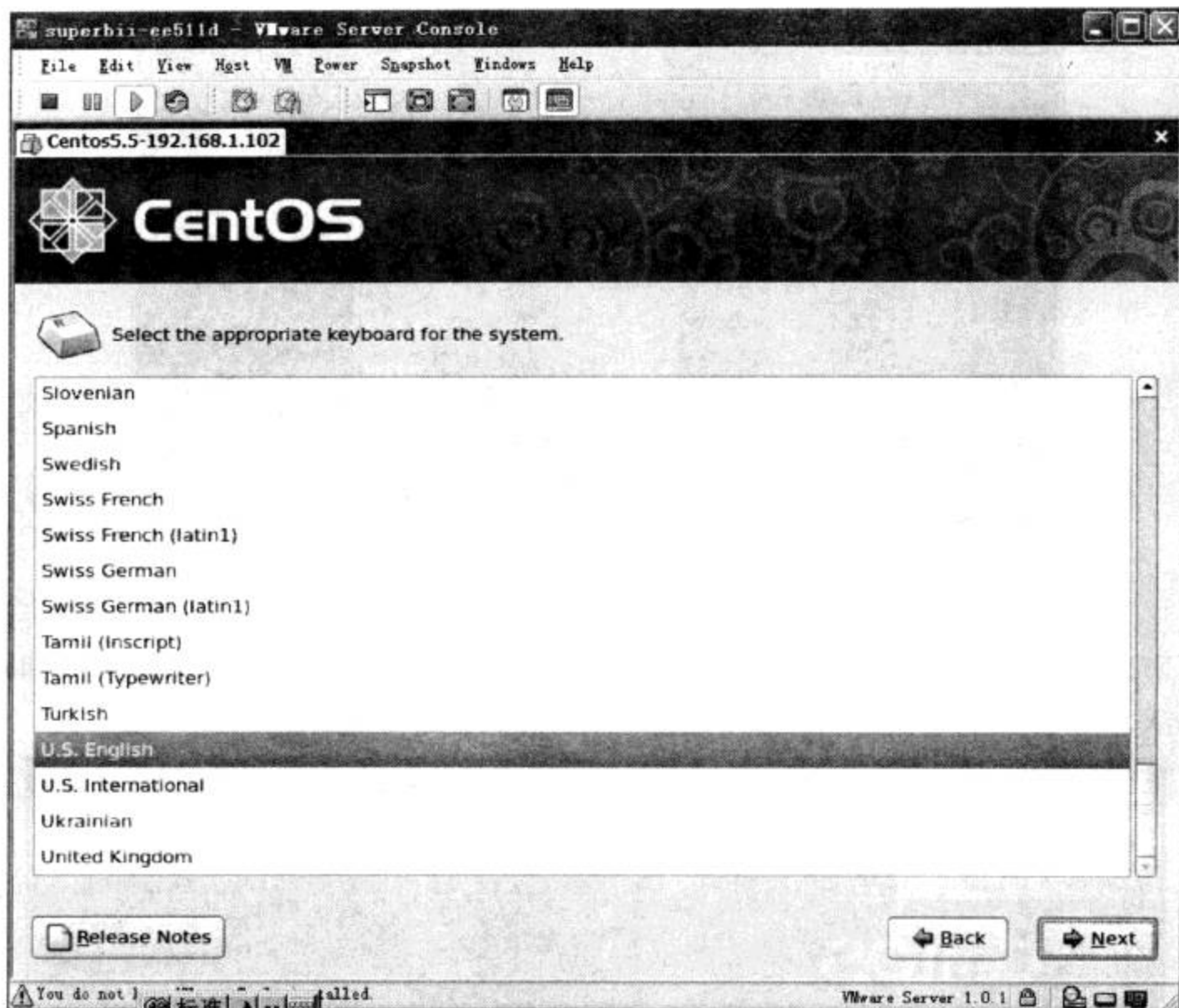


图 1-7 选择安装哪种键盘

图 1-8 所示表示系统发现了一块 10 237MB 的设备，它在向你请示是否需要安装这块硬盘并清空所有数据，这里选择 YES（当然了，如果在实际工作中进行选择就要小心了，它会删除所有的硬盘数据）。

图 1-9 所示是选择分区方式，建议不用 Centos5.5 自带的 Disk Druid 将硬盘的所有分区都分成 LVM（逻辑磁盘卷），虽然 LVM 可以在生产系统上直接在线扩展硬盘的分区，但一般应用的机器都没有此需求（有此特殊应用需求的机器除外），所以这里选择自定义分区。在机房安装 Centos 时，一般是没有鼠标的，大家可以用 Alt + 快捷键来操作，免得用 Tab 键一步步切换，以节约时间（此快捷键在重装系统删除分区时非常管用，另一个快捷键也非常管用，即 Shift 键加上 Tab 键，它会回退一步，这个操作可以简化我们的工作。它

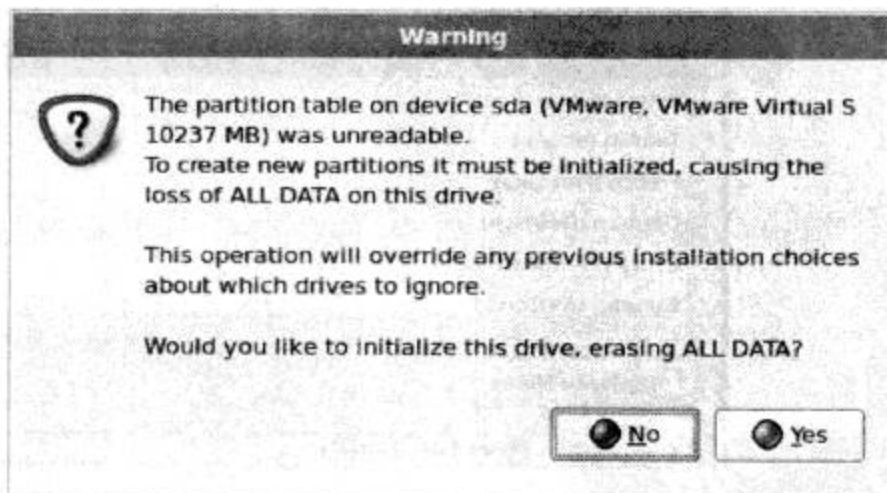


图 1-8 是否选择删除所有的数据

们在 Windows 系统上一样适用)。

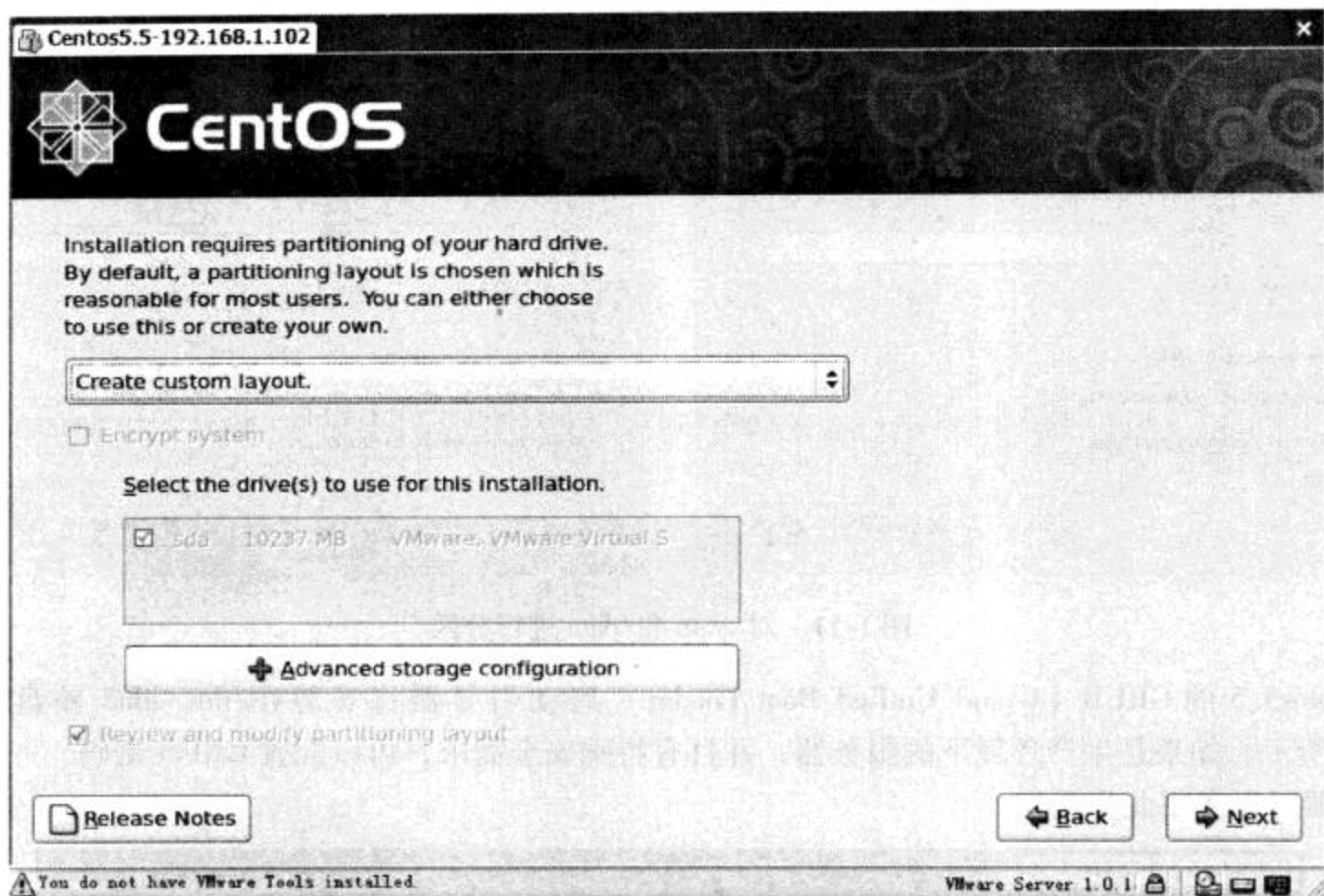


图 1-9 选择硬盘的分区方式

Centos5.5 x86_64 的 /boot 比较小，我们一般选择 100MB 左右即可，也可以根据实际环境来选择，最好不超过 10GB，如图 1-10 所示。

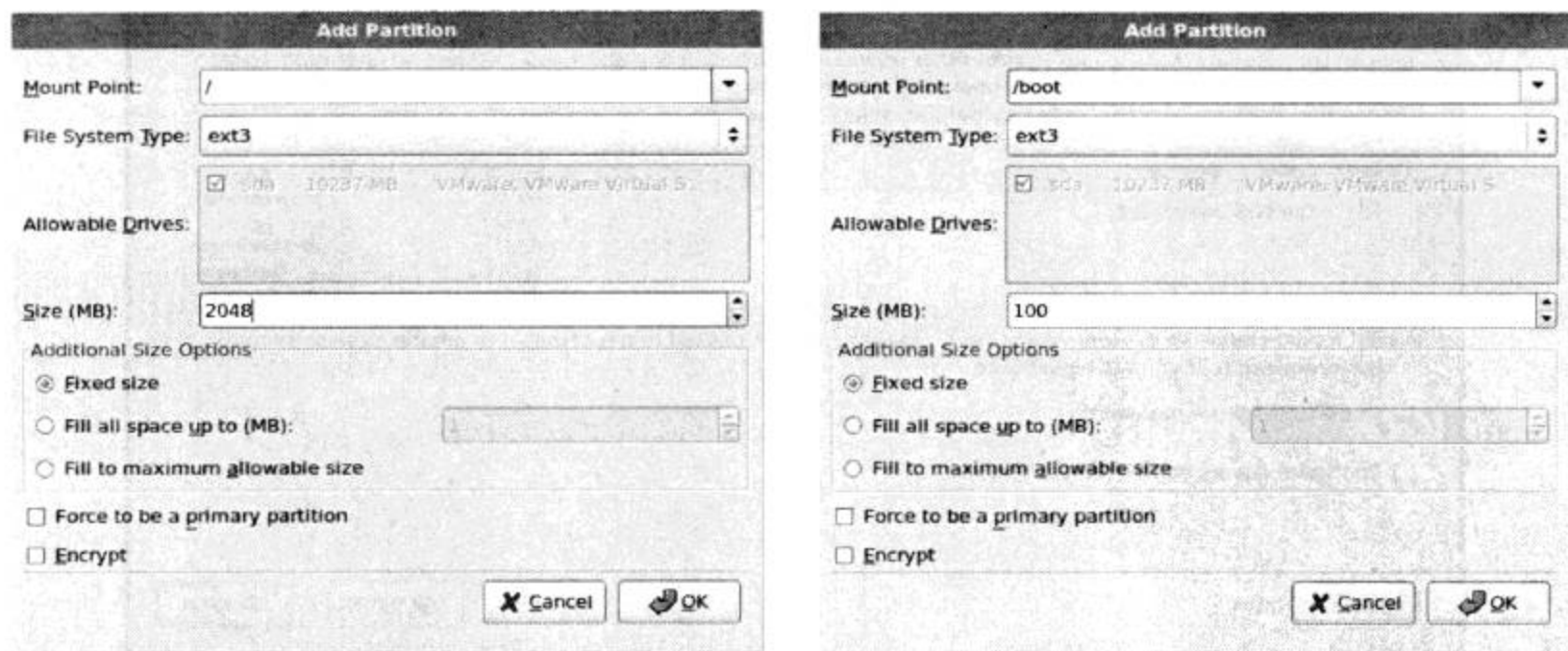


图 1-10 对 / 和 /boot 进行分区

关于 swap 分区，我们一般选择内存的 1.5 ~ 2 倍，超过 2 倍就没有什么意义了。我这里的内存只有 384MB，所以我选择 512 这个数值。如果是 8GB 内存的机器，那就适合用 12 000MB，如图 1-11 所示。

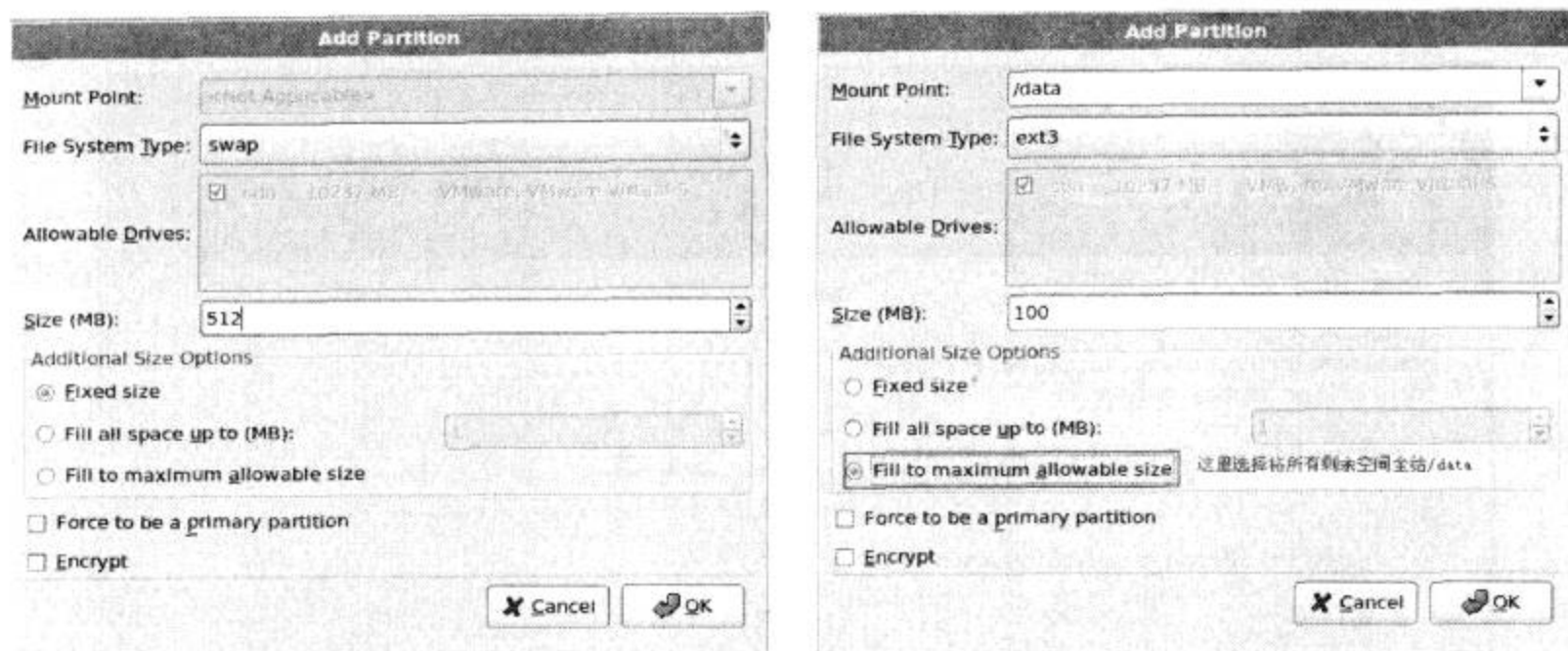


图 1-11 对 swap 和 /data 进行分区

Centos5.5 的 GRUB (Grand Unified Boot Loader) 启动引导器将安装在 /dev/sda2 硬盘上, 如图 1-12 所示。如果是生产环境下的服务器, 并且有特殊安全需求, 可以配置 GRUB 密码, 一般的应用服务器可以省略此步操作。

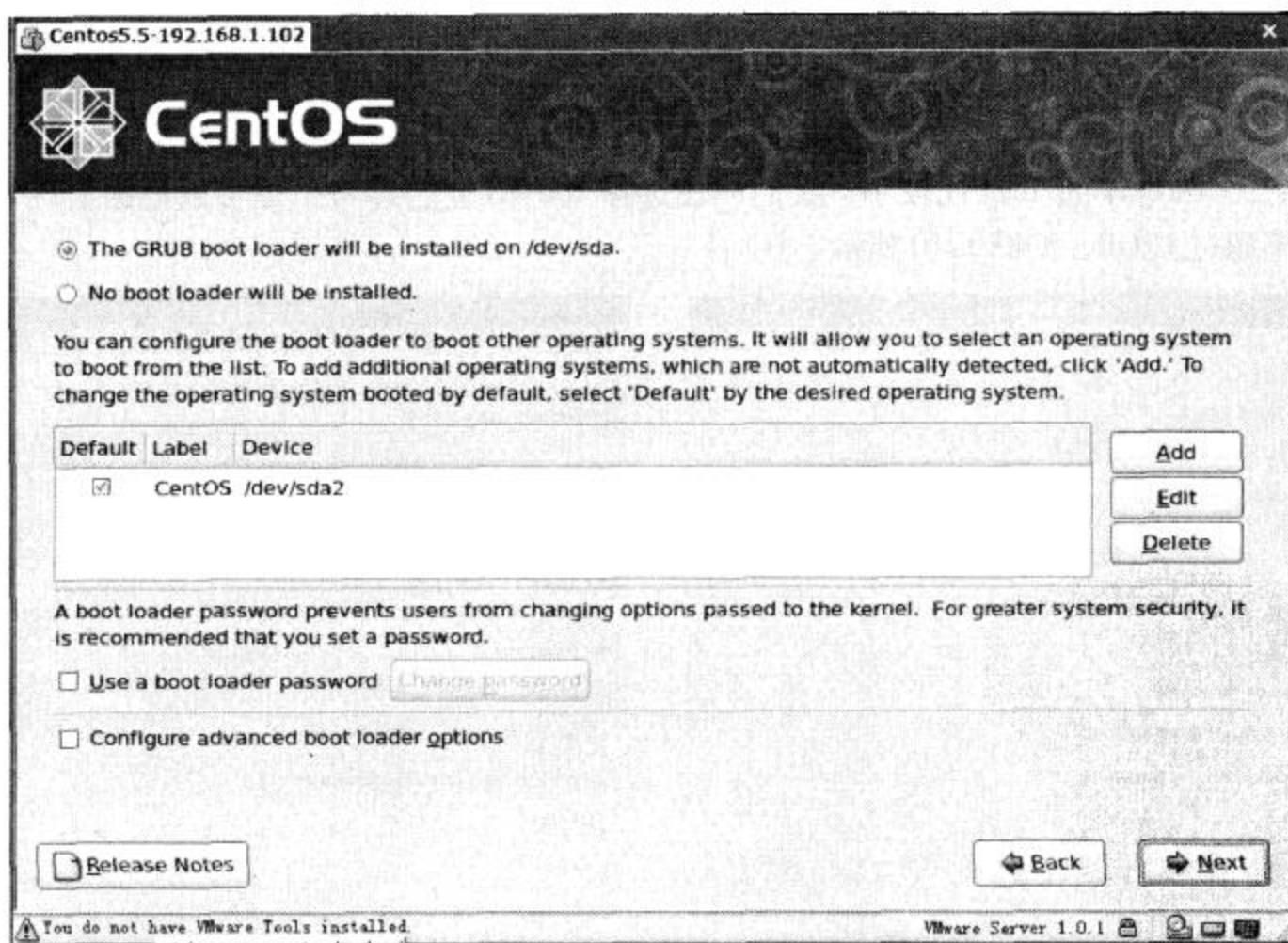


图 1-12 GRUB 的安装位置以及是否配置 GRUB 密码

配置系统的网卡时, 记得选上 “Active on Boot”, 如图 1-13 所示。如果不选择, 会直接导致系统在启动时网卡分配不了 IP, 这也是我们经常易犯的错误之一。

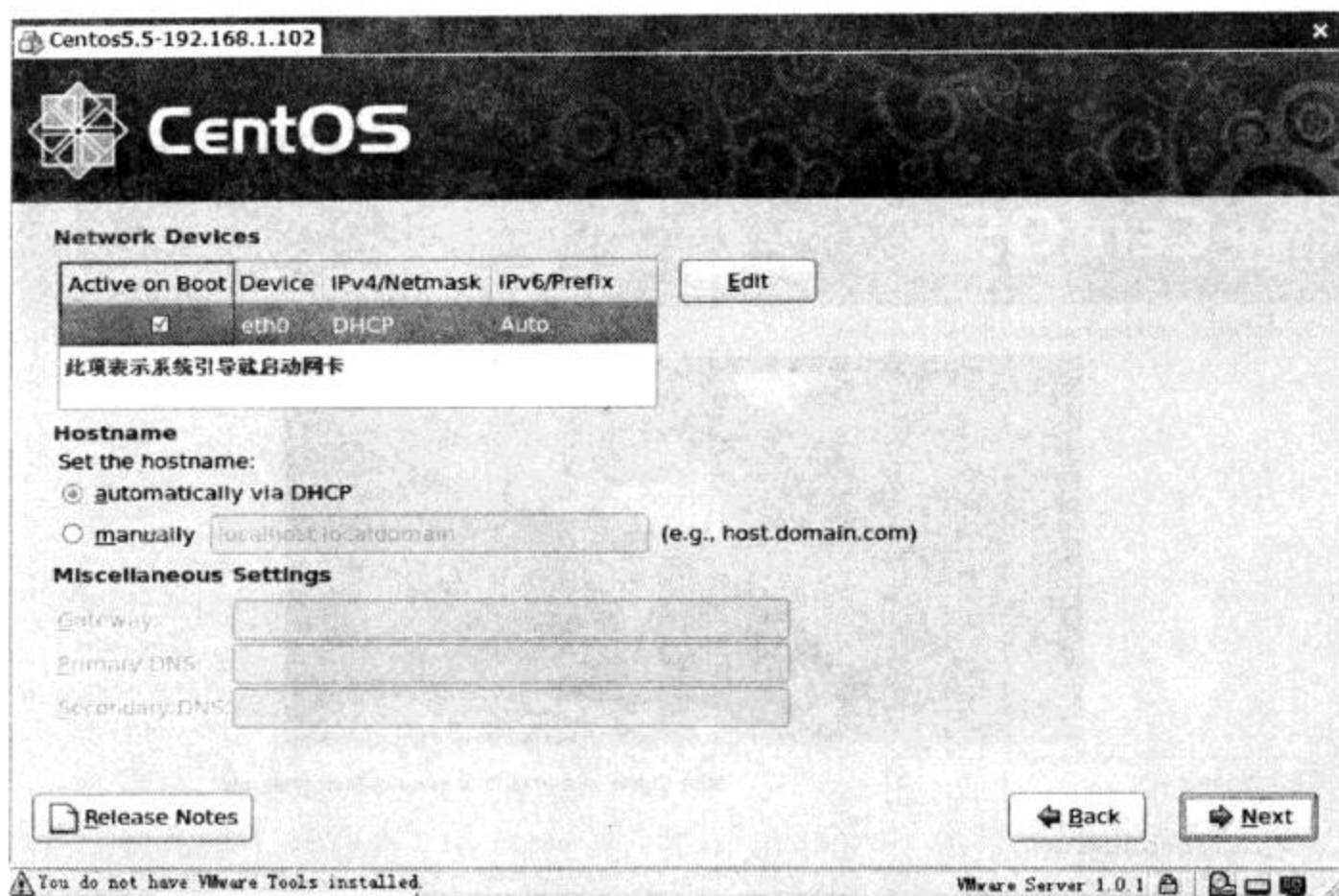


图 1-13 选择是否在系统启动时引导网卡

接下来，就进入了网卡配置界面。每个系统的网络配置大同小异，一般就是 IP 地址、子网掩码、网关及主从 DNS 服务器等。Centos5.5 的网卡配置也比较直观，如图 1-14 所示。

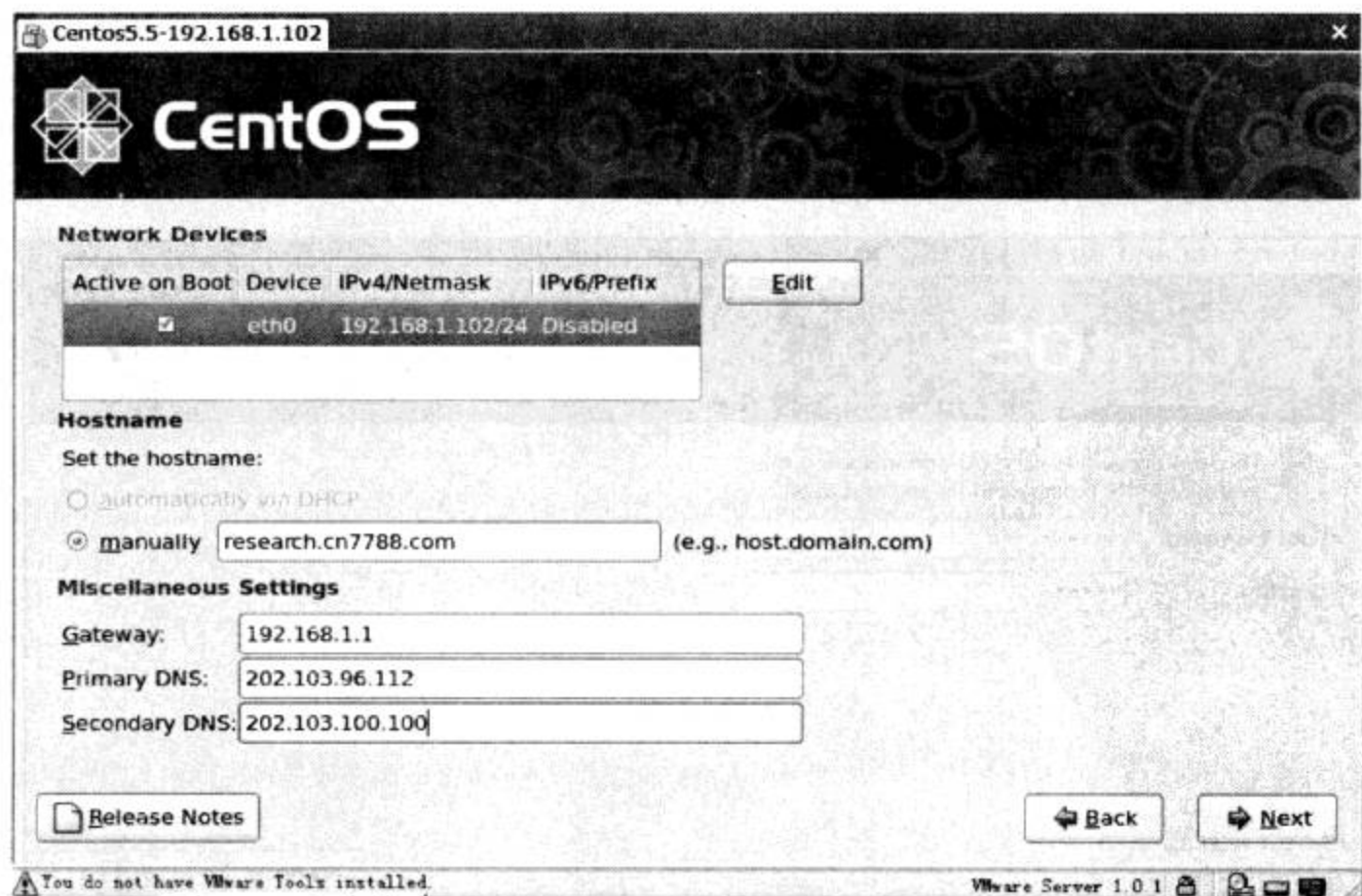


图 1-14 网卡配置界面

图 1-15 所示是进行系统时区的选择，一般选择“亚洲/上海”。大家稍微注意看一下，这里有一个“System clock uses UTC”，这个选项中的 UTC 是什么意思呢？这里简单解释一下。

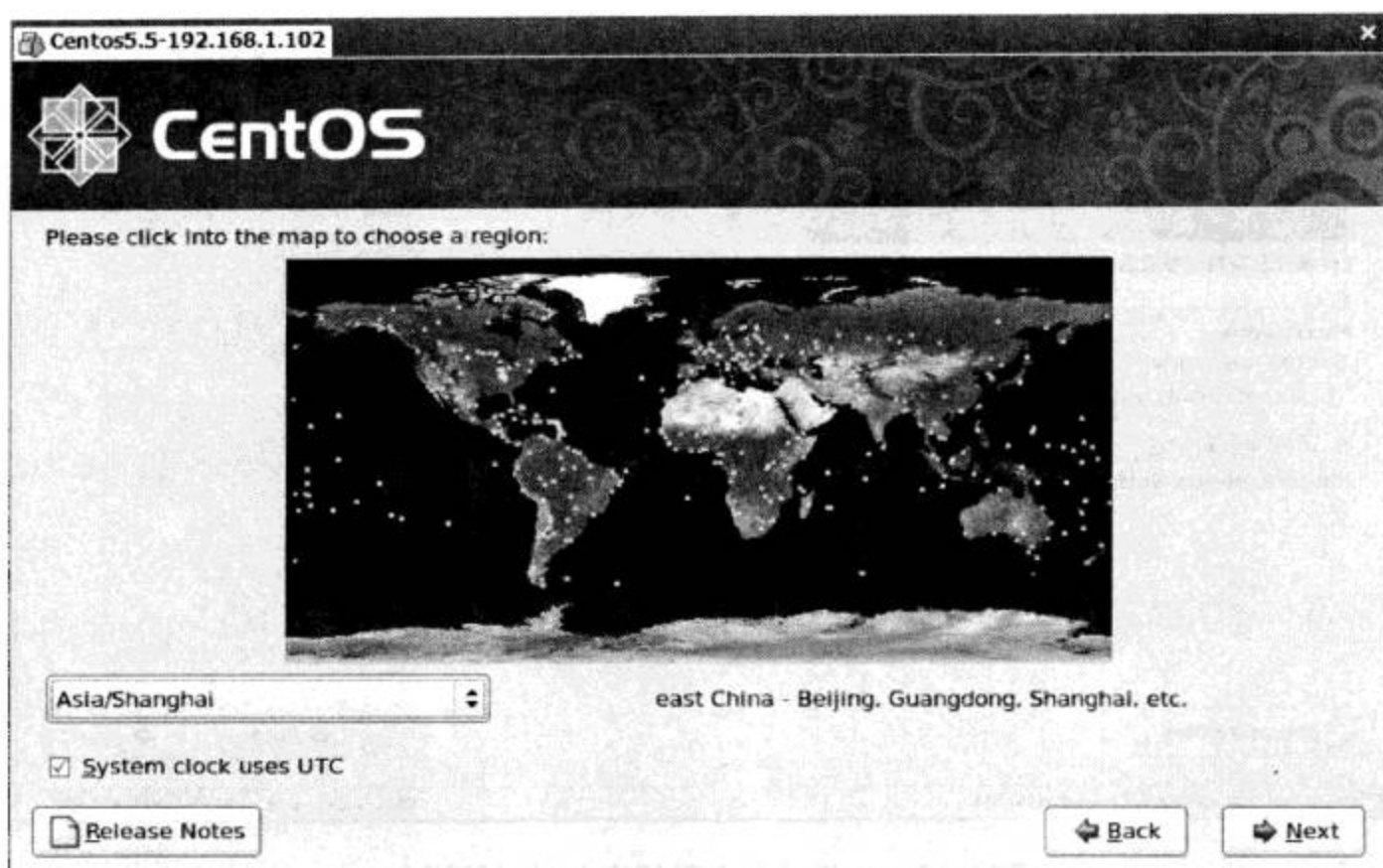


图 1-15 为 Centos 系统选择时区

UTC 表示世界协调时间 (Universal Time Coordinated)。GPS 系统中有两种时间区分，一为 UTC，另一为 CST (地方时)。两者的区别在于时区不同，UTC 指的是 0 时区的时间，地方时为本地时间，如北京为早上 8 点 (东八区)，UTC 时间就为零点，时间比北京时间晚 8 个小时，以此推算即可。这里建议大家勾选此项。

接下来就该设置 Root 的密码了，如图 1-16 所示。

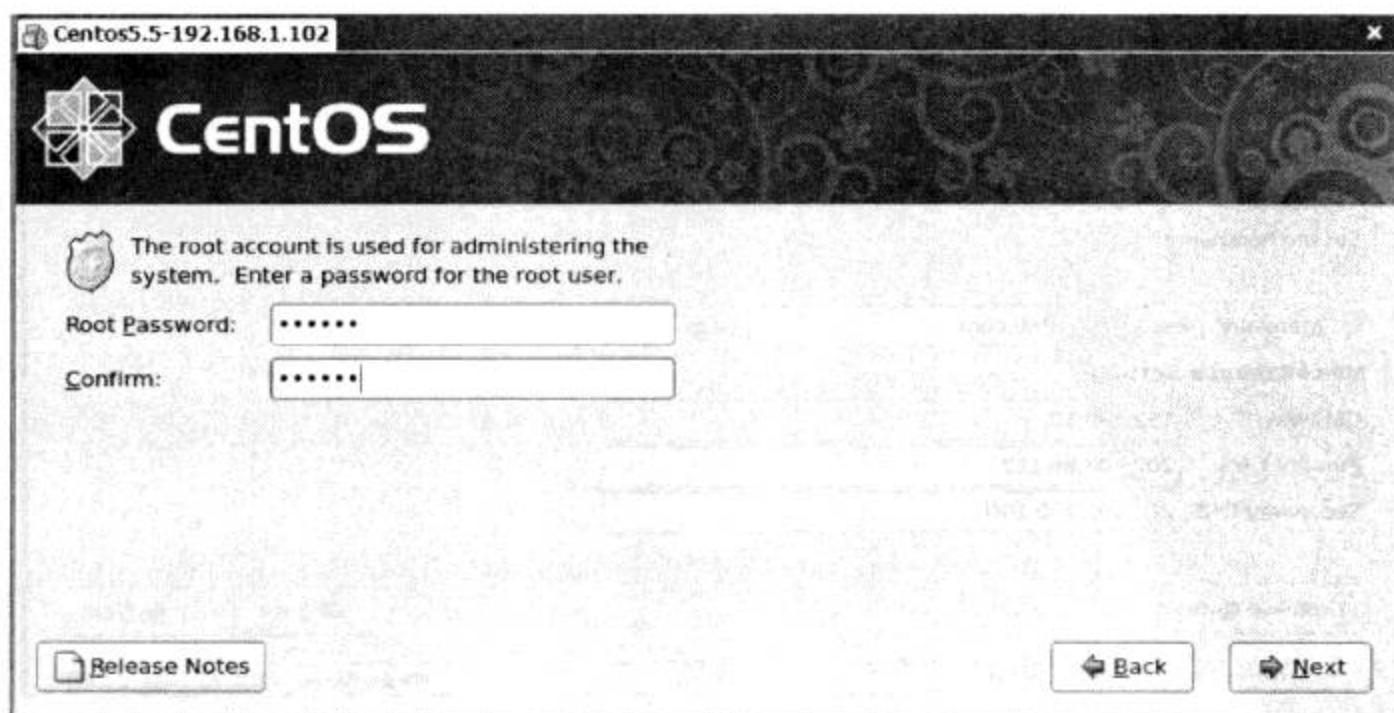


图 1-16 Root 密码设置

然后安装系统的软件包。这里提醒一下大家，我们在安装线上环境的系统时，要牢记一个原则，软件包越少的系统越稳定。这里我们仅仅选择 Server，即最小化安装，如图 1-17 所示。不过，如果服务器有特殊用途，则另当别论了。选择好后就进入了图 1-18 所示的 Centos5.5 准备安装界面。

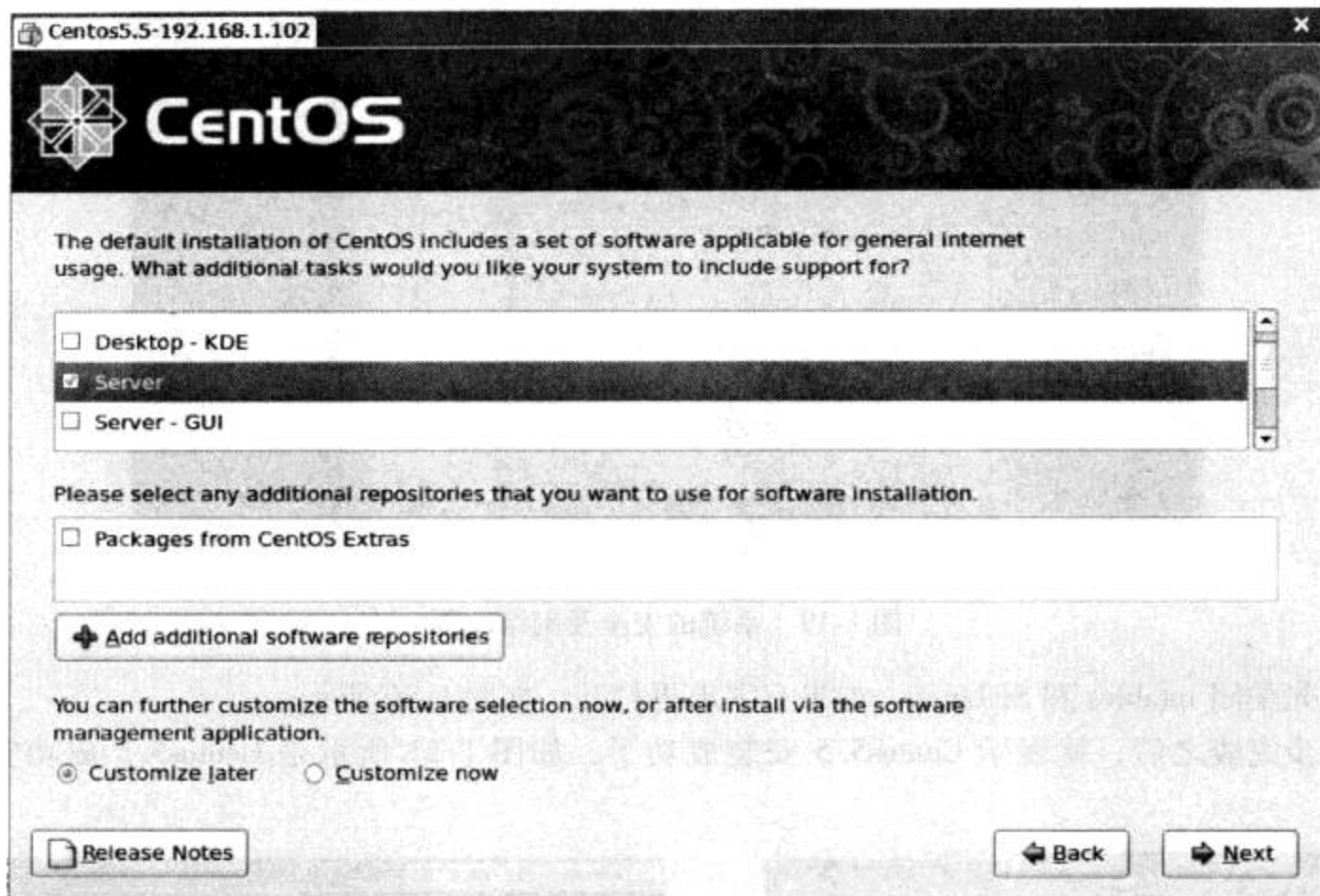


图 1-17 选择系统的软件包

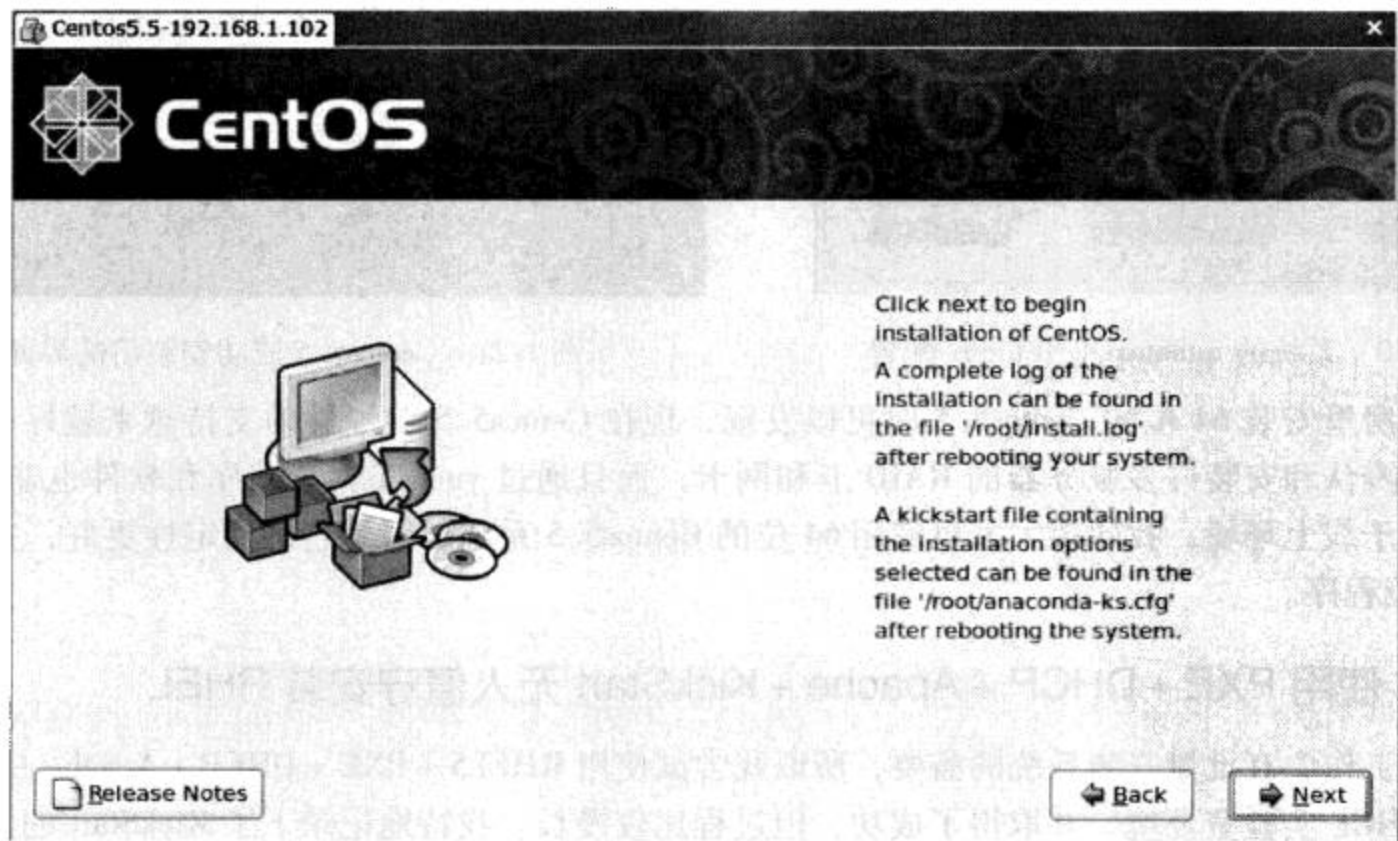


图 1-18 准备安装 Centos 系统界面

系统安装完成后重启。接着就要选择系统的安全及网络配置了，这里我们主要关注“Firewall configuration”这个选项，其他选项默认即可，如图 1-19 所示。

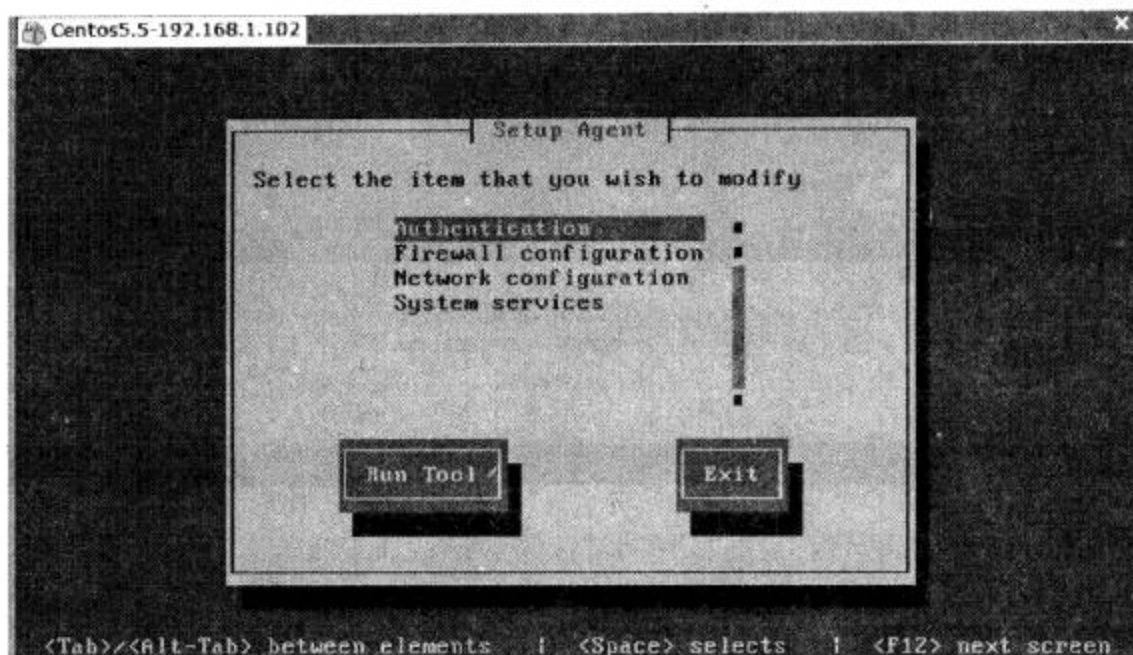


图 1-19 系统的安全及网络配置

暂时先关闭 iptables 和 SELinux，如果有需求再打开，如图 1-20 所示。

这一步完成之后，就表示 Centos5.5 安装成功了，如图 1-21 所示是 Centos5.5 成功安装后的界面。

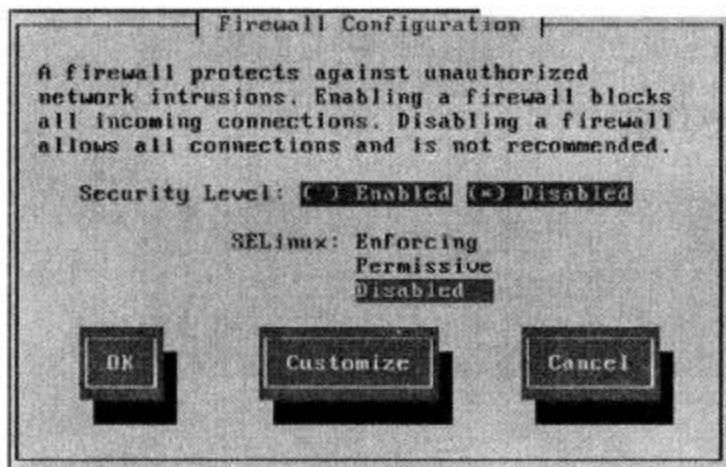


图 1-20 系统的 iptables 和 SELinux 配置



图 1-21 Centos5.5 成功安装后的界面

在机房里安装 64 位的 Centos5.5 时可以发现，现在 Centos5.5 对硬件的支持越来越好了，它能很轻松地辨认和安装许多服务器的 RAID 卡和网卡，而且通过 yum 安装基础库和软件也非常方便。即使是用于线上环境，我建议大家都使用 64 位的 Centos5.5 系统，因为它的稳定性更好，并且向下兼容 32 位程序。

1.1.2 使用 PXE + DHCP + Apache + KickStart 无人值守安装 RHEL

因为工作中有批量安装系统的需要，所以我尝试使用 RHEL5 + PXE + DHCP + Apache + KickStart 安装了 RHCE 实验室环境，并取得了成功，但过程比较漫长。我特地记录下了 KickStart 的无人值守安装过程。现阶段，此方法主要用于在公司内批量安装大量的新服务器系统，然后上架到公网 IDC 机房，这极大地简化了用光盘重复安装 Centos5.5 的过程，提高了工作效率。以下安装方法在 32 位

的 RHEL 5.1/5.2/5.3 下也可以通过，当然同样适用于 32 位和 64 位的 Centos5.5 系列了。

首先，我们来介绍一下与之相关的原理和概念。

1. 什么是 PXE

严格来说，PXE 并不是一种安装方式，而是一种引导方式。进行 PXE 安装的必要条件是在要安装的计算机中必须包含一个 PXE 支持的网卡（NIC），即网卡中必须要有 PXE Client。PXE（Pre-boot Execution Environment）协议可以使计算机通过网络启动。此协议分为 Client 端和 Server 端，而 PXE Client 则在网卡的 ROM 中。当计算机引导时，BIOS 把 PXE Client 调入内存中执行，然后由 PXE Client 将放置在远端的文件通过网络下载到本地运行。运行 PXE 协议需要设置 DHCP 服务器和 TFTP 服务器。DHCP 服务器会给 PXE Client（将要安装系统的主机）分配一个 IP 地址，由于是给 PXE Client 分配 IP 地址，所以在配置 DHCP 服务器时需要增加相应的 PXE 设置。此外，在 PXE Client 的 ROM 中，已经存在了 TFTP Client，那么它就可以通过 TFTP 协议到 TFTP Server 上下载所需的文件了。

2. 什么是 KickStart

KickStart 是一种无人值守的安装方式。它的工作原理是在安装过程中记录典型的需要人工干预填写的各种参数，并生成一个名为 ks.cfg 的文件。如果在安装过程中（不只局限于生成 KickStart 安装文件的机器）出现要填写参数的情况，安装程序首先会去查找 KickStart 生成的文件，如果找到合适的参数，就采用所找到的参数；如果没有找到合适的参数，便需要安装者手工干预了。所以，如果 KickStart 文件涵盖了安装过程中可能出现的所有需要填写的参数，那么安装者完全可以只告诉安装程序从何处取 ks.cfg 文件，然后就去忙自己的事情。等安装完毕，安装程序会根据 ks.cfg 中的设置重启系统，并结束安装。

3. PXE + KickStart 的安装条件和详细步骤

执行 PXE + KickStart 安装需要的设备为：

- ☐ DHCP 服务器
- ☐ TFTP 服务器
- ☐ KickStart 所生成的 ks.cfg 配置文件
- ☐ 一台存放系统安装文件的服务器，如 NFS、HTTP 或 FTP 服务器
- ☐ 一个带有 PXE 支持网卡的主机

安装的具体步骤如下。

1) 安装 httpd，代码如下所示：

```
yum -y install httpd*
```

2) 挂载 RHEL5.1 的 DVD 光盘，并复制光盘下的所有内容（文件和文件夹）到 /var/www/html 下。无论是 RHEL 还是 Centos 的光盘，如果是最小化安装，基本上第一张 DVD 光盘就足够了，挂载及复制的代码如下所示：

```
mount /dev/cdrom /mnt
cp -rf /mnt/* /var/www/html
```

3) 安装 tftp-server，并启用 tftp 服务，同时启动 xinetd 进程，代码如下所示：

```
1.rpm -ivh tftp-server-0.39-1.i386.rpm
```



```
2.vi /etc/xinetd.d/tftp
# default: off
# description: The tftp server serves files using the trivial file transfer\
# protocol. The tftp protocol is often used to boot diskless\
# workstations, download configuration files to network-aware printers,\
# and to start the installation process for some operating systems.
service tftp
{
    socket_type = dgram
    protocol = udp
    wait = yes
    user = root
    server = /usr/sbin/in.tftpd
    server_args = -s /tftpboot
    disable = no #disable 的值由 yes 变为 no
    per_source = 11
    cps = 100 2
    flags = IPv4
}
3.service xinetd restart
```

4) 配置支持 PXE 的启动程序（注意：前面已经把第一张光盘的内容复制到/var/www/html 目录中了，所以需要文件时只要从/var/www/html 目录中复制就行了。但是在这里描述的时候，我还是标明了文件的真实位置，比如在 DVD 光盘的哪个文件中，便于大家理解）。

□ 建立 tftpboot 文件夹，如下所示（若该文件夹已经存在则不用建立）：

```
mkdir -p tftpboot
```

□ 复制 pxelinux.0 文件至 tftpboot 文件夹中，如下所示：

```
cp /usr/lib/syslinux/pxelinux.0 /tftpboot
```

□ 把 Linux 第一张安装光盘上的/image/pxeboot/initrd.img 和 vmlinux 复制到/tftpboot/中，如下所示：

```
cp /var/ftp/image/pxeboot/initrd.img /tftpboot
cp /var/ftp/image/pxeboot/vmlinux /tftpboot
```

□ 复制第一张安装光盘上的 isolinux/*.msg 到/tftpboot 目录下，如下所示：

```
cp /var/ftp/isolinux/*.msg /tftpboot
```

□ 在 tftpboot 中新建一个 pxelinux.cfg 目录如下：

```
mkdir pxelinux.cfg
```

□ 把第一张安装光盘上 isolinux 目录中的 isolinux.cfg 复制到 pxelinux.cfg 目录中，同时更改文件名称为 default，代码如下：

```
cd pxelinux.cfg
cp /var/ftp/isolinux/isolinux.cfg /tftpboot/pxelinux.cfg/default
```

□ 在上一个步骤中，暂时不要修改 default 文件，进行到这一步时，虽然已经可以通过网络来

引导并手动安装 KickStart 了,但是由于这里讨论的是无人值守安装,所以先不修改这个 default 文件。

5) 安装 DHCP 服务,同时修改如下配置:

```
rpm -ivh dhcp-3.0.1-12_EL.i386.rpm
```

然后复制配置模板文件到指定的目录中,并重新命名:

```
cp /usr/share/doc/dhcp-3.0.1/dhcpd.conf.sample /etc/dhcpd.conf
```

接着修改配置文件,添加一行: filename "/pxelinux.0" (注意写入的位置,不然会导致安装失败,切记!)。其他的修改大家可以根据如下代码完成:

```
[root@localhost isolinux]# vim /etc/dhcpd.conf
ddns-update-style interim;
ignore client-updates;
next-server 192.168.1.14;      #PXE 服务器 IP 地址
filename "/pxelinux.0";      #注意此行的位置,如果写在 subnet 下面,安装会失败

subnet 192.168.1.0 netmask 255.255.255.0 {
# --- default gateway
    option routers                192.168.1.254;
    option subnet-mask            255.255.255.0;
    option nis-domain             "example.com";
    option domain-name            "example.com";
    option domain-name-servers    192.168.1.254;

    option time-offset            -18000; # Eastern Standard Time
#   option ntp-servers            192.168.1.1;
#   option netbios-name-servers   192.168.1.1;
# --- Selects point-to-point node (default is hybrid). Don't change this unless
# -- you understand Netbios very well
#   option netbios-node-type 2;
    range dynamic-bootp 192.168.1.128 192.168.1.254;
    default-lease-time 21600;
    max-lease-time 43200;
    # we want the nameserver to appear at a fixed address
    #host ns {
        #   next-server marvin.redhat.com;
        #   hardware ethernet 12:34:56:78:AB:CD;
        #   fixed-address 192.168.1.110;
    #}
}
```

最后启动 DHCP 服务如下:

```
service dhcpd start
```

6) 安装 KickStart,同时配置 KickStart。

□ 首先需要安装 KickStart 这个工具包,在 RHEL 最小化安装系统时,此软件包并没有默认安装。命令如下所示:

```
rpm -ivh system-config-kickstart-2.5.16-2.noarch.rpm
```

□ 在 gnome 环境下配置 KickStart。命令如下所示：

```
system-config-kickstart
```

运行上面的命令后可以对系统的一些基本配置进行设置，例如选择时区、设置 Root 的密码等。

□ 接下来便要进行安装了，建议选择 httpd 安装，在安装过程中，根据引导选择安装选项，不需要做更改。

□ KickStart 会让我们选择需要批量安装的 RHEL 分区信息，按照上一节所介绍的 Centos5.5 x86_64 的安装方法，我们创建 4 个分区，即 /、/boot、/data 和 swap 分区。

□ 在进行网络配置时，我使用的是静态分配地址（动态同样如此），这里跟前面进行光盘安装是一样的。

□ 设置显示配置时可以按照我们的习惯选择。

□ 关于软件包的选择，大家可以根据实际的工作需求来选择自己需要的软件包，尽量不要同时安装 Kernel Development 和 Development Tools，不然的话在安装的时候很容易出错。

□ 其他都选择默认设置，不需要修改。

□ 最后将生成的文件 ks.cfg 保存到 /var/www/html 下。

7) 修改 /tftpboot/pxelinux.cfg/default 文件，指定读取 ks.cfg 的方法（ks = http://192.168.1.40/ks.cfg）。

8) vim /var/www/html/ks.cfg，代码如下所示：

```
auth --useshadow --enablemd5
key --skip #一定要有这一行,用来跳过注册码输入,不然安装会失败
bootloader --location=mbr
clearpart --all --initlabel
text
firewall --disabled
firstboot --disable
keyboard us
lang en_US
logging --level=info
url --url=http://192.168.1.14/
network --bootproto=dhcp --device=eth0 --onboot=on
reboot
rootpw --iscrypted $1$HEJKfwF9$r1l0JoPz74ToF9NbE3Qs1
selinux --disabled
timezone --isUtc Asia/Shanghai
intall
xconfig --defaultdesktop=GNOME --depth=8 --resolution=640x480
part swap --bytes-per-inode=4096 --fstype="swap" --size=512
part /boot --bytes-per-inode=4096 --fstype="ext3" --size=200
part / --bytes-per-inode=4096 --fstype="ext3" --grow --size=1
%packages
@cluster-storage
@mysql
@development-libs
@editors
```

```

@text - internet
@x - software - development
@virtualization
@legacy - network - server
@dns - server
@gnome - desktop
@dialup
@core
@base
@ftp - server
@network - server
@clustering
@java - development
@base - x
@chinese - support
@web - server
@smb - server
@printing
@admin - tools
@development - tools
@graphical - internet
kmod - gnbd - XEN
kmod - gfs - XEN
perl - Convert - ASN1
perl - Crypt - SSLeay
mesa - libGLU - devel
tftp - server
kexec - tools
bridge - utils
device - mapper - multipath
ypserv
openldap - servers
vnc - server
dhcp
xorg - x11 - server - Xnest
xort - x11 - server - Xvfb
imake
gcc - objc
expect

```

9) 重新引导安装, 问题解决了, 最终安装成功。

以上是我的配置步骤, 如果完全照此操作, 是一定可以成功安装的! 本节内容我在自己的博客里也有发表过, 很多朋友按照此方法配置成功了。他们也提出许多宝贵的意见, 本节内容也是在此基础上修改而成的。

1.1.3 Linux 的其他安装方法

Linux 的安装方法还有许多, 比如说通过 ISO 硬盘安装。还有许多系统管理员非常喜欢用 U 盘, 在服务器上自动安装系统, 不过所用的原理与上述原理基本相同。但这些方法没有前面介绍的两种方法通用。在实际工作中, 光盘安装和网络无人值守安装属于主流安装方法, 特别是网络无人值守

安装方式，适合大规模部署集群环境的服务器。大家可根据公司的需求来安装系统，怎么方便怎么来，毕竟高效率也是我们追求的目标之一。

1.2 全面了解 Linux 服务器

作为一名项目实施工程师，我经常要到客户的机房或托管的 IDC 从事相关的系统工作，里面的 Linux 服务器少则十几台，多则上百台。为了便于工作，通常这时候我就会非常清楚地了解服务器的硬件配置。但是客户往往并不能提供详细的硬件配置清单，仅仅能提供系统管理员的密码。所以我总结了一些查看服务器情况的命令，它们可以很清楚地反映 Linux 服务器的状态、性能等详细信息。以下内容完全出自项目的实践总结，也希望大家能够熟练掌握，以方便自己工作。

1.2.1 查看 Linux 服务器的 CPU 详细情况

判断 Linux 服务器 CPU 情况的依据如下：

- 具有相同 core id 的 CPU 是同一个 core 的超线程。（Any cpu with the same core id are hyper-threads in the same core.）
- 具有相同 physical id 的 CPU 是同一个 CPU 封装的线程或核心。（Any cpu with the same physical id are threads or cores in the same physical socket.）

下面以笔者自己的 PowerEdge 2850 为例进一步进行说明。

物理 CPU 个数如下所示：

```
[root@localhost ~]# cat /proc/cpuinfo | grep "physical id" | sort | uniq | wc -l
2
```

每个物理 CPU 中 core 的个数（即核数）如下所示：

```
[root@localhost ~]# cat /proc/cpuinfo | grep "cpu cores" | uniq
cpu cores          : 1
```

逻辑 CPU 的个数如下所示：

```
[root@localhost ~]# cat /proc/cpuinfo | grep
"processor" | wc -l
4
```

另外，在 Windows 2003 中，逻辑 CPU 的个数可以通过任务管理器（调出命令是 taskmgr）“性能”下的“CPU 使用”情况看出。如图 1-22 所示是我的 AMD 双核速龙 5500+ 的“CPU 使用”情况。

其实大家从这里就可以看出来，按理说物理 CPU 个数 × 核数就应该等于逻辑 CPU 的个数，如果不相等的话，则表示你的服务器 CPU 支持超线程技术。我们在配置服务器的应用时，应以服务器的逻辑 CPU 的个数为准。

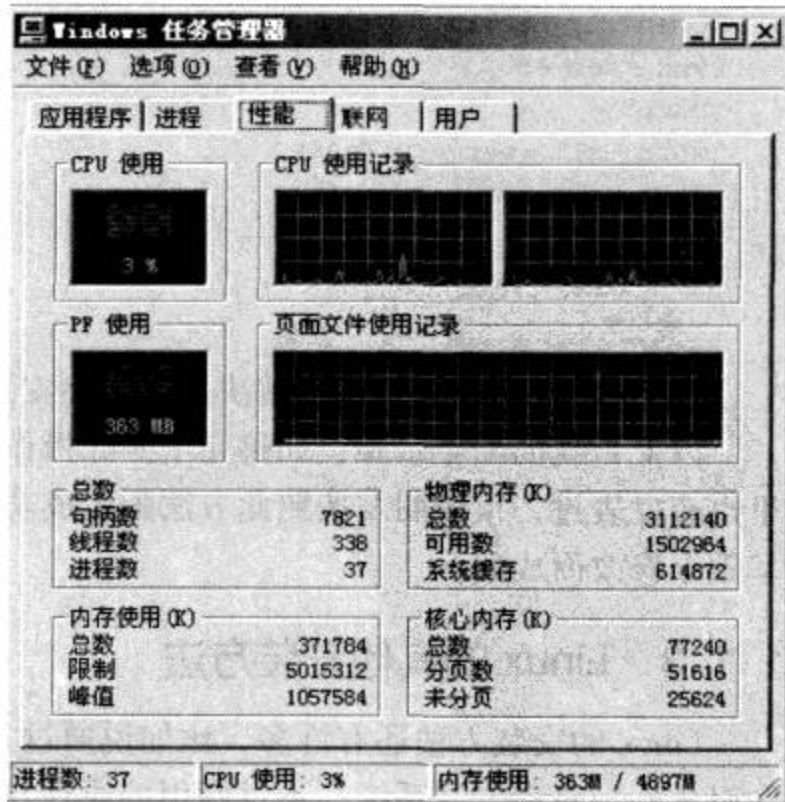


图 1-22 Windows 2003 下的任务管理器

1.2.2 查看 Linux 服务器的内存使用情况

查看 Linux 服务器下的内存使用情况，可以使用命令 `free -m`。注意此命令只在 Linux 下有效，在 FreeBSD 中没有此命令。命令如下所示：

```
[root@localhost ~]# free -m
total      used      free      shared    buffers     cached
Mem:       3949      1397      2551         0        268       917
-/+ buffers/cache:      211      3737
Swap:      8001         0      8001
```

- total: 内存总数
- used: 已经使用的内存数
- free: 空闲的内存数
- shared: 多个进程共享的内存总额
- -buffers/cache: (已用) 的内存数，即 `used-buffers-cached`
- +buffers/cache: (可用) 的内存数，即 `free + buffers + cached`

得出结论：

可用内存的计算公式为：

可用内存 = free + buffers + cached，即 2551MB + 268MB + 917MB = 3737MB

很久以前在笔记本上用 Ubuntu8.04 时就觉得 Linux 管理内存的机制非常优秀，简而言之：Linux 的内存是拿来用的，而不是拿来看的。我与一个朋友探讨 Linux 的使用情况时，他问我为什么 Linux 使用的内存这么高。他机器上 1GB 的内存 free 才 232MB，而 Windows XP 才用了 200MB 不到的样子。这其实是被 Linux 的 free 命令之表象迷惑了，Linux 的内存使用是很有讲究的。还是举例说明，如下的 free 命令所显示的是当前内存的使用情况，-m 的意思是用 M 个字节来显示内容，我们来一起看看。

```
#free -m
total used free shared buffers cached
Mem: 1002 769 232 0 62 421
-/+ buffers/cache: 286 715
Swap: 1153 0 1153
```

在第一部分 Mem 行中有如下参数。

- total: 内存总数，即 1002MB
- used: 已经使用的内存数，即 769MB
- free: 空闲的内存数，即 232MB
- shared: 当前已经废弃不用，总是 0
- buffers Buffer: 缓存内存数，即 62MB
- cached Page: 缓存内存数，即 421MB

其中，内存总数与已使用内存数和空闲内存数的关系是：

total (1002M) = used (769M) + free (232M)

在第二部分内容（-/+ buffers/cache）中各参数如下所示。

- （- buffers/cache）：used 内存数，即 286MB（指的是第一部分 Mem 行中的 used- buffers- cached）。
- （+ buffers/cache）：free 内存数，即 715MB（指的是第一部分 Mem 行中的 free + buffers + cached）。

可见 -buffers/cache 反映的是被程序实实在在用掉的内存，而 + buffers/cache 反映的是可以挪用的内存总数。

第三部分是指交换（swap）分区，大家应该都明白，这里就不再讲了。

有可能大家看了上面的解释还是不太明白。比如：第一部分（Mem）与第二部分（-/+ buffers/cache）的结果有关，used 和 free 为什么这么奇怪？其实我们可以从两个方面来分析。对操作系统来讲这两项是 Mem 的参数，buffers/cached 都属于被使用，所以它认为 free 只有 232MB；对应用程序来讲 + buffers/cached 等同于可用的内存，因为 buffer/cached 可提高程序执行的性能，当程序使用内存时，buffer/cached 很快就会被使用。所以从应用的角度来看，应以（-/+ buffers/cache）的 free 和 used 为主，即我们主要看与它相关的 free 和 used 就可以了。另外告诉大家一些常识，为了提高磁盘和内存的存取效率，对 Linux 做了很多精心的设计，除了对 dentry 进行缓存（用于 VFS、加速文件路径名到 inode 的转换）外，还采取了两种主要 Cache 方式：Buffer Cache 和 Page Cache。前者用于针对磁盘块的读写，后者用于针对文件 inode 的读写。这些 Cache 能有效地缩短 I/O 系统调用（比如 read、write、getdents）的时间。

在 Linux 中，内存是拿来用的，不是拿来看的。而在 Windows 中，无论你的真实物理内存有多少，它都会用硬盘交换文件来读，即使是内存还有一大部分。这也就是 Windows 常常提示虚拟空间不足的原因。可以想见，硬盘怎么会快过内存，所以我们在观察 Linux 的内存使用情况时，只要没发现用 swap 的交换空间，就不用担心自己的内存太少。如果常常看到 swap 用了很多，那么你就要考虑加物理内存了。这也是在 Linux 服务器上看内存是否够用的标准。

1.2.3 查看 Linux 服务器的硬盘使用情况

想要了解 Linux 服务器的硬盘使用情况，可以根据以下步骤来查看。

1) 查看硬盘及分区信息，如下所示：

```
[root@localhost ~]# fdisk -l
Disk /dev/sda: 146.5 GB, 146548981760 bytes
255 heads, 63 sectors/track, 17816 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1  *           1           13       104391   83   Linux
/dev/sda2             14          6387     51199155   83   Linux
/dev/sda3          6388          7407      8193150   82   Linux swap / Solaris
/dev/sda4          7408         17816     83610292+   5   Extended
/dev/sda5          7408         17816     83610261   83   Linux
```

以上表明这是一块 0.15TB 的服务器硬盘。

2) 检查文件系统的磁盘空间占用情况，如下所示：

```
[root@localhost ~]# df -h
```



```
avg-cpu: %user   %nice %system %iowait  %steal   %idle
          0.00    0.00    0.00    0.00    0.00  100.00
```

Device:	rrqm/s	wrqm/s	r/s	w/s	rsec/s	wsec/s	avgrq - sz	avgqu - sz	await	svctm
%util										
sda	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
sda1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
sda2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
sda3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
sda4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
sda5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

```
avg-cpu: %user   %nice %system %iowait  %steal   %idle
          0.00    0.00    0.00    0.00    0.00  100.00
```

Device:	rrqm/s	wrqm/s	r/s	w/s	rsec/s	wsec/s	avgrq - sz	avgqu - sz	await	svctm
%util										
sda	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
sda1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
sda2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
sda3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
sda4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
sda5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

参数说明:

- rrqm/s: 每秒进行 merge 的读操作数目, 即 $\text{delta}(\text{rmerge})/\text{s}$ 。
- wrqm/s: 每秒进行 merge 的写操作数目, 即 $\text{delta}(\text{wmerge})/\text{s}$ 。
- r/s: 每秒完成的读 I/O 设备的次数, 即 $\text{delta}(\text{rio})/\text{s}$ 。
- w/s: 每秒完成的写 I/O 设备的次数, 即 $\text{delta}(\text{wio})/\text{s}$ 。
- rsec/s: 每秒读扇区数, 即 $\text{delta}(\text{rsect})/\text{s}$ 。
- wsec/s: 每秒写扇区数, 即 $\text{delta}(\text{wsect})/\text{s}$ 。
- kB/s: 每秒读 K 字节数。是 rsec/s 的一半, 因为每扇区大小为 512 字节。
- kB/s: 每秒写 K 字节数。是 wsect/s 的一半。
- avgrq-sz: 平均每次设备 I/O 操作的数据大小 (即扇区), 即 $\text{delta}(\text{rsect} + \text{wsect})/\text{delta}(\text{rio} + \text{wio})$ 。
- avgqu-sz: 平均 I/O 队列的长度, 即 $\text{delta}(\text{aveq})/\text{s}/1000$ (除以 1000 是因为 aveq 的单位为毫秒)。
- await: 平均每次设备 I/O 操作的等待时间 (单位: 毫秒), 即 $\text{delta}(\text{ruse} + \text{wuse})/\text{delta}(\text{rio} + \text{wio})$ 。
- svctm: 平均每次设备 I/O 操作的服务时间 (单位: 毫秒), 即 $\text{delta}(\text{use})/\text{delta}(\text{rio} + \text{wio})$ 。
- %util: 一秒中有百分之多少的时间用于 I/O 操作, 或者说一秒中有多少时间 I/O 队列是非空的, 即 $\text{delta}(\text{use})/\text{s}/1000$ (因为 use 的单位为毫秒)。

大家看着这些参数和名词解释可能不太明白, 其实在工作中我们只关注以下几个方面即可:

- 如果%util 接近 100%，说明产生的 I/O 请求太多，I/O 系统已经满负荷，该磁盘可能存在瓶颈。
- 如果 idle 小于 70%，I/O 的压力就比较大了，说明读取进程中有较多的 wait。同时还可以结合 vmstat 查看 b 参数（等待资源的进程数）和 wa 参数（I/O 等待所占用的 CPU 时间的百分比，高过 30% 时 I/O 的压力就比较高了），帮助了解 I/O 的负荷情况。

另外还可以参考如下一些情况：svctm 应小于 await，因为同时等待请求的等待时间被重复计算了。一般来说，svctm 的大小和磁盘性能有关，CPU/内存的负荷也会对其有一定的影响，请求过多就会间接导致 svctm 增加。

await 的大小一般取决于服务时间（svctm）以及 I/O 队列的长度和 I/O 请求的发出模式。如果 svctm 比较接近 await，说明 I/O 几乎没有等待时间；如果 await 远大于 svctm，说明 I/O 队列太长，应用得到的响应时间也变长。如果响应时间超过了用户允许的范围，这时就可以考虑更换更快的磁盘、调整内核 elevator 的算法、优化应用或者升级 CPU。

队列长度（avgqu-sz）也可作为衡量系统 I/O 负荷的指标，但由于 avgqu-sz 是按照单位时间计算出来的平均值，所以不能反映瞬间的 I/O 洪水。

4) 查看 Linux 系统中某目录的大小，这在工作中经常会遇到。我们可以用命令：du-sh 目录名来查看，如下所示：

```
[root@localhost /]# du -sh /root
1.2M /root
```

检查是否有分区使用率（Use%）过高（比如超过 90%）的情况，如发现某个分区空间接近用完，可以进入该分区的挂载点，用以下命令找出占用空间最多的文件或目录，然后按照从大到小的顺序，找出系统中占用最多空间的前 10 个文件或目录。

```
# du -du -cks * | sort -rn | head -n 10
```

5) dd 命令在 Linux 系统中也经常用到，很多时候维护系统工作时需要使用到它。用 dd 命令可以把指定的输入文件拷贝到指定的输出文件中，并且在拷贝过程中可以进行格式转换。我碰到的几个需要用到 dd 命令的地方如下。

□ 制作交换文件的时候，例如：

```
dd if=/dev/zero of=/swapfile bs=1024 count=65536
```

□ 制作驱动盘的时候，例如，将硬盘上的驱动文件拷贝到一个软驱中：

```
dd if=rhel40.img of=/dev/fd0 bs=10k 或者
dd if=mptlinux-3.02.68-1-rhel4.i686.dd of=/dev/fd0 bs=10k
```

□ 制作 ISO 镜像的时候，例如：

```
dd if=/dev/cdrom of=/root/cd1.iso
```

当然也可以用 mkisofs 命令来制作 ISO 镜像。

另外，重装系统的时候，我喜欢用 dd 命令来破坏系统的分区表。由于此命令破坏性极大，会带来严重的后果，为了防止读者误操作，这里就不介绍了，有兴趣的读者可自行研究。

下面将分别介绍 dd 命令的参数，如下：

```
if = file
```

输入文件名，默认为标准输入。

```
of = file
```

输出文件名，默认为标准输出。

```
ibs = bytes
```

一次读入 bytes 个字节（即一个块大小为 bytes 个字节）。

```
obs = bytes
```

一次写 bytes 个字节（即一个块大小为 bytes 个字节）。

```
bs = bytes
```

同时设置读写块的大小为 bytes，可代替 ibs 和 obs。

```
cbs = bytes
```

一次转换 bytes 个字节，即转换缓冲区大小。

```
skip = blocks
```

从输入文件开头跳过 blocks 个块后再开始复制。

```
seek = blocks
```

从输出文件开头跳过 blocks 个块后再开始复制（通常当输出文件介质是磁盘或磁带时才有效）。

```
count = blocks
```

仅拷贝 blocks 个块，块大小等于 ibs 指定的字节数。

1.2.4 查看 Linux 系统的平均负载

1. Linux 系统的平均负载的概念

有时候我们会觉得系统响应很慢，但是又找不到原因，这时就要查看平均负载了，看它是否有大量的进程在排队等待。特定时间间隔内运行队列中的平均进程数可以反映系统的繁忙程度，所以我们通常会在自己的网站或系统变慢时第一时间查系统的负载，即 CPU 的平均负载。

2. 查看平均负载

究竟应该如何查看平均负载呢？最简单的命令是 `uptime`，如下所示：

```
[root@localhost ~]# uptime
11:31:11 up 11 days, 19:01, 2 users, load average: 0.02, 0.01, 0.00
```

目前的主流服务器都是双四核，有相当强悍的 CPU，做一般的应用服务的话，Linux 系统的负载这块倒不用我们担心。

还可以用 `w` 命令来查看，顺便可以查看一下系统当前有哪些用户，他们占用了哪些终端，如下所示：

```
[root@localhost ~]# w
```

```
11:33:00 up 11 days, 19:03, 2 users, load average: 0.00, 0.00, 0.00
USER      TTY      FROM          LOGIN@   IDLE   JCPU   PCPU WHAT
root      pts/1    113.57.224.3   09:03    2:11m  0.04s  0.04s -bash
root      pts/2    113.57.224.3   11:31    0.00s  0.02s  0.00s w
```

另外，还有动态命令 `top`，这个命令也可以反映系统负载情况。在下面的命令提示中，我们只关心加粗字体部分。

```
[root@localhost ~]# top
top - 11:37:47 up 11 days, 19:08, 2 users, load average: 0.00, 0.00, 0.00
Tasks: 122 total, 1 running, 121 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.1%us, 0.0%sy, 0.0%ni, 99.9%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 4044136k total, 1435504k used, 2608632k free, 274740k buffers
Swap: 8193140k total, 0k used, 8193140k free, 941884k cached
```

上面加粗字体显示的内容是什么意思呢？再通过 `uptime` 查看一下。

```
[root@localhost ~]# uptime
11:39:36 up 11 days, 19:16, 1 user, load average: 0.09, 0.03, 0.01
```

原来它所表示的是过去的 1 分钟、5 分钟和 15 分钟内进程队列中的平均进程数量。

那么，如何衡量当前系统是否负载过高呢？可以从以下几点来考虑。

- 如果每个 CPU（可以按 CPU 核心的数量计算）当前的活动进程数不大于 3，则系统性能良好。
- 如果每个 CPU 当前的活动进程数不大于 4，表示可以接受。
- 如果每个 CPU 当前的活动进程数大于 5，则系统性能问题严重。

还可以结合 `vmstat` 命令来判断我们的系统是否过于繁忙，如果确定很繁忙的话，就要考虑是否更换服务器或增加 CPU 的个数了。总结如下：

如果 `r` 经常大于 3 或 4，且 `id` 经常少于 50，则表示 CPU 的负荷很重。

在上面例子中，我的服务器是 PowerEdge 2850，CPU 是双核双线程的，则 $0.09/2 = 0.045$ （即负载值/真实 CPU 个数），此系统的 CPU 负载基本可以忽略了。事实上，现在主流服务器的 CPU 都很强悍，如果不是应用虚拟化等特殊场景，基本上负载都很小。

按照前面的计算公式，我所配置 Nagios 报警的 CPU 负载阈值为 CPU 核心的数量（即 CPU 的物理个数 × 核数）。还是以我的服务器 PowerEdge 2850 为例，其 CPU 核心的数量为 $2 \times 2 = 4$ ，则设置报警值为 4。这样设置是合理的，因为毕竟不是每个应用服务器的 CPU 都支持多核心，毕竟整个网站中还有些性能比较弱的服务器是用来做备份的。

1.2.5 查看 Linux 系统的其他参数^①

1. 用 `vmstat` 来监控 Linux 系统的整体性能

`vmstat` 是一个相当全面的性能分析工具，可以用来观察系统的进程状态、内存使用情况、虚拟内存的使用情况、磁盘的 I/O、中断、上下文切换、CPU 的使用情况等性能信息。建议熟练掌握此命令。举例如下：

^① 注：本节内容不仅适用于 CentOS 5.5，也适用于其他 Linux 系统。

```
[root@localhost ~]# vmstat 1 4
```

```
procs -----memory----- --swap-- -----io----- --system-- -----cpu-----
r  b  swpd  free    buff    cache  si  so  bi  bo  in   cs   us  sy  id   wa  st
0  0  0      2251592  343104  741248 0   0   0   2   3    10   0   0  100   0   0
0  0  0      2251592  343104  741248 0   0   0   0  1034  193   0   0  100   0   0
0  0  0      2251592  343104  741248 0   0   0   0  1017  147   0   0  100   0   0
0  0  0      2251592  343104  741248 0   0   0   0  1028  183   0   0  100   0   0
```

其中：

□ procs

r: 等待运行的进程数。

b: 处在非中断睡眠状态的进程数。

w: 被交换出去的可运行的进程数。此数由 Linux 计算得出，但 Linux 并不耗尽交换空间。

□ memory

swpd: 虚拟内存使用情况，单位为 KB。

free: 空闲的内存，单位为 KB。

buff: 被用来作为缓存的内存数，单位为 KB。

□ swap

si: 从磁盘交换到内存的交换页数量，单位为 KB。

so: 从内存交换到磁盘的交换页数量，单位为 KB。

□ io

bi: 发送到块设备的块数，单位为块。

bo: 从块设备接收到的块数，单位为块。

□ system

in: 每秒的中断数，包括时钟中断。

cs: 每秒的环境（上下文）切换次数。

□ cpu

按 CPU 的总使用百分比来显示。

us: CPU 使用时间。

sy: CPU 系统使用时间。

id: 闲置时间。

标准情况下 r 和 b 值应该为：

$r < 5$, $b \approx 0$

假设输出的信息中：

□ r 经常大于 3 或 4，且 id 经常少于 50，表示 CPU 的负荷很重。

□ bi、bo 长期不等于 0，表示内存不足。

□ disk 经常不等于 0，且在 b 中的队列大于 2 或 3，表示 io 的性能不好。

2. 查看系统内核

查看系统内核主要是为了掌握其版本号，为安装 LVS 等软件做准备。我们可以用命令 `uname -a` 来查看，如下所示：


```
[root@localhost ~]# uname -a
Linux localhost.localdomain 2.6.18-194.el5 #1 SMP Fri Apr 2 14:58:14 EDT 2010 x86_64 x86_64 x86_64
GNU/Linux
```

简化的参数命令如下：

```
[root@localhost ~]# uname -r
2.6.18-194.el5
```

如果要查看系统是 32 位还是 64 位，可以用如下命令：

```
[root@localhost /]# ls -lF / | grep /$
```

此命令会查找是否有/lib64 的目录，有则表示系统为 64 位，无则表示系统为 32 位。大家记住一点，64 位的 CPU 系统架构可以安装 32 位或 64 位的系统，而 32 位的 CPU 架构只能安装 32 位的系统。查找情况如下所示：

```
drwxr-xr-x 2 root root 4096 03-13 04:02 bin/
drwxr-xr-x 4 root root 1024 03-08 16:44 boot/
drwxr-xr-x 5 root root 4096 03-27 00:58 data/
drwxr-xr-x 11 root root 3800 03-17 07:27 dev/
drwxr-xr-x 101 root root 12288 03-26 08:47 etc/
drwxr-xr-x 4 root root 4096 03-09 10:34 home/
drwxr-xr-x 11 root root 4096 03-13 04:02 lib/
drwxr-xr-x 7 root root 4096 03-13 04:02 lib64/
drwx----- 2 root root 16384 03-08 16:33 lost+found/
drwxr-xr-x 2 root root 4096 2010-01-27 media/
drwxr-xr-x 2 root root 0 03-16 16:23 misc/
drwxr-xr-x 2 root root 4096 2010-01-27 mnt/
drwxr-xr-x 2 root root 0 03-16 16:23 net/
drwxr-xr-x 2 root root 4096 2010-01-27 opt/
dr-xr-xr-x 142 root root 0 03-16 16:22 proc/
drwxr-xr-x 17 root root 4096 03-28 11:30 root/
drwxr-xr-x 2 root root 12288 03-13 04:02 sbin/
drwxr-xr-x 2 root root 4096 03-08 16:35 selinux/
drwxr-xr-x 2 root root 4096 2010-01-27 srv/
drwxr-xr-x 11 root root 0 03-16 16:23 sys/
drwxrwxrwt 5 root root 4096 03-28 04:02 tmp/
drwxr-xr-x 15 root root 4096 03-08 16:40 usr/
drwxr-xr-x 21 root root 4096 03-08 16:47 var/
```

另一种常见方法是通过 file 命令来判断系统中的文件是 32 位还是 64 位的，以此作为判断系统的依据，如下所示：

```
[root@localhost /]# file /sbin/init
/sbin/init: ELF 64-bit LSB executable, AMD x86-64, version 1 (SYSV), for GNU/Linux 2.6.9, dynamically
linked (uses shared libs), for GNU/Linux 2.6.9, stripped
```

此结果表示系统为 64 位的。

3. 查看服务器使用的 Linux 发行版的相关信息

下面的命令可查看服务器使用的 Linux 发行版的名称、版本号及描述信息等：

```
[root@localhost /]# lsb_release -a
LSB Version: :core-3.1-ia32:core-3.1-noarch:graphics-3.1-ia32:graphics-3.1-noarch
Distributor ID: CentOS
Description: CentOS release 5.5 (Final)
Release: 5.5
Codename: Final
```

如果 Centos5.5 或以前的版本没有此命令，我们可以通过 `yum -y install redhat-lsb` 来安装。

4. 查看系统已载入的相关模块

Linux 操作系统的核心具有模块化的特性，因此在编译核心时，无须把全部的功能都放入核心。可以将这些功能编译成一个个单独的模块，待需要时再分别载入。比如说在安装 LVS + Keepalived 时，我们经常会用 `lsmod` 来查看 lvs 模块是否已经载入，如下所示：

```
root@localhost ~]# lsmod | grep ip_vs
ip_vs_wrr          35905  1
ip_vs              122113  3 ip_vs_wrr
```

5. 在 Linux 下查找 PCI 设置

有时需要在 Linux 下查找 PCI 设置。这时可以用 `lspci` 命令，它能列出机器中的 PCI 设备信息，比如声卡、显卡、Modem、网卡等的信息，主板集成设备的信息也能列出来。`lspci` 读取的是 `hwdata` 数据库。可能有读者和我一样，最关心的还是网卡型号。

```
[root@localhost ~]# lspci | grep Ether
06:07.0 Ethernet controller: Intel Corporation 82541GI Gigabit Ethernet Controller (rev 05)
07:08.0 Ethernet controller: Intel Corporation 82541GI Gigabit Ethernet Controller (rev 05)
```

网卡的监控一般用命令 `miit-tool` 和 `iptraf`，这个知识点将在后面讲解。

本节主要从服务器的 CPU、内存、硬盘性能、负载及其他方面详细说明了 Linux 服务器的整体性能状态，希望大家能够通过以上所列的方法来了解自己的 Linux 服务器的性能状态，这对工作会有很大帮助。

1.3 Linux 服务器的网络配置

服务器的系统安装好后，接下来就要在机房或内网环境中配置它的 IP 了，这是最重要的一个环节。下面我就以 64bit Centos5.5 服务器为例来说明如何通过命令或图形来配置 Linux 服务器的 IP、网关、DNS，以及如何用命令查看 Linux 的进程或网络连接等。

1.3.1 配置 Linux 服务器的网络

1. 手动修改配置网卡文件

手动配置网卡是最直接的方式，熟练的系统管理员在平时维护系统的时候更喜欢使用这种方式，因为手动配置有如下优点：

- 熟悉命令之后，手动配置更快速，并且不需要重新启动。
- 能够使用配置命令的高级特性。

- 更容易维护配置文件，找出系统故障。
- 能更深刻地了解系统配置是如何进行的。

那么，下面就介绍一下如何手动配置网卡文件。首先检查网卡是否正常安装，Centos5.5 的驱动非常强悍，基本上市面上的服务器网卡都可以正常安装，我们可以用如下命令检查网卡是否正常安装：

```
[root@localhost ~]# lspci |grep Ether
06:07.0 Ethernet controller: Intel Corporation 82541GI Gigabit Ethernet Controller (rev 05)
07:08.0 Ethernet controller: Intel Corporation 82541GI Gigabit Ethernet Controller (rev 05)
[root@localhost ~]# dmesg |grep error
```

一般来说，2.4 版本以后的 Linux 可以支持的网卡芯片组驱动已经很完备了，包括著名厂商（如 Intel），以及使用广泛的 RealTek、Via 等网卡芯片，所以大家可以很轻松地使用它们的网卡。我们还可以用 `lsmod` 命令通过加载模块的方法来加载特殊的网卡。

在配置 Linux 网络设备时，它们分别被赋予别名，该别名由一个描述性的缩略词和一个编号组成。第一个设备的编号为 0，其他设备依次为 1、2、3……其中，`eth0`、`eth1` 是以太网卡接口，大多数的以太网卡都用此名表示，包括许多并行端口以太网卡，接下来主要讨论这种类型的网卡。与网卡相关的 TCP/IP 网络配置文件是 `/etc/sysconfig/network-scripts/ifcfg-ethx`，其中 `x` 是从 0 开始的，第一个以太网配置文件即 `/etc/sysconfig/network-scripts/ifcfg-eth0`。以我的公网机器举例说明如下：

```
[root@localhost ~]# vim /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
BOOTPROTO=none
HWADDR=00:14:22:1B:71:20
IPV6INIT=yes
IPV6_AUTOCONF=yes
ONBOOT=yes
NETMASK=255.255.255.192
IPADDR=203.93.236.146
GATEWAY=203.93.236.129
TYPE=Ethernet
PEERDNS=yes
USERCTL=no
NETMASK=255.255.255.192
IPADDR=203.93.236.146
```

其中：

- `DEVICE=eth0` 表示设定网卡的名称，它要跟文件名称对应。
- `BOOTPROTO=none` 是启动时 IP 取得的协议，这里是固定的（此值也可以为 `static`），如果是动态主机的话，要改成 `dhcp`。
- `HWADDR=00:14:22:1B:71:20` 指网卡的 MAC 地址，可以用 `ifconfig` 来取值。当然了，如果我们不指定这项的话，Centos5.5 也会默认指定。
- `IPV6INIT=yes` 表示支持 IPv6，`no` 表示不支持。
- `IPV6_AUTOCONF=yes` 表示自动配置 IPv6。
- `ONBOOT=yes` 表示在开机的时候启动网卡。这里肯定要选择 `yes` 了，如果选择 `no` 的话则网

卡在系统引导时不会被分配 IP 地址，那就很麻烦了。

- `NETMASK = 255.255.255.192` 和 `IPADDR = 203.93.236.146`，这两个就没什么好说了，这是我们的 IDC 分配给公网的 IP 地址和子网掩码，强悍的是，顺序反了一样生效。
- `GATEWAY = 203.93.236.129` 是网关地址。
- `TYPE = Ethernet` 表示网卡的类型为以太网型。
- `PEERDNS = yes` 表示允许从 DHCP 获得的 DNS 覆盖本地的 DNS。
- `USERCTL = no` 表示不允许普通用户修改配置。

配置完成后记得保存，然后重启服务 `server network restart` 即可生效。当然了，如果嫌麻烦，可以用 Centos5.5 的 `setup` 工具中的“网络配置”来操作，方法很简单，如图 1-23 所示。这里就不浪费篇幅了。

2. 修改机器的 hostname

下面来修改机器的 hostname，如下所示：

```
vim /etc/sysconfig/network
NETWORKING=yes
NETWORKING_IPV6=yes
HOSTNAME=localhost.localdomain
```

`HOSTNAME` 后面紧跟的就是我们的主机名，这里是系统默认的 `localhost.localdomain`。

`HOSTNAME` 的后面即可接我们要更改的主机名，重启后可以用 `hostname` 命令来查看。如果只是简单地用命令 `hostname`，仅仅对当前生效，重启后会失效，比较好的方法是写到文件中保存。

3. 修改主机名查询静态表/etc/hosts

Linux 主机名的相关配置文件就是 `/etc/hosts`，这个文件告诉本主机哪些域名对应哪些 IP，哪些主机名对应哪些 IP。下面对 `/etc/hosts` 的格式进行说明。一般 `/etc/hosts` 的内容会与下面的内容类似：

```
127.0.0.1 localhost.localdomain localhost
192.168.21.100 webserver.cn7788.com webserver
192.168.21.111 ftp.cn7788.com ftp
```

通常 `hosts` 文件的每行为一个主机的信息，并且每行由 3 部分组成，各个部分间由空格隔开，这三部分所表示的意思如下。

第一部分：网络 IP 地址

第二部分：主机名或域名

第三部分：主机名别名

当然每行也可以是两部分，即主机 IP 地址和主机名，比如：

```
192.168.21.100 webserver.cn7788.com
```

另外，`hosts` 文件中以 `#` 号开头的行是说明，不会被系统解释。

这里稍微解释一下主机名（hostname）和域名（domain）的区别：主机名通常在局域网内使用，通过 `hosts` 文件，主机名就被解析到对应的 IP 地址上；域名通常在 Internet 上使用，但如果本

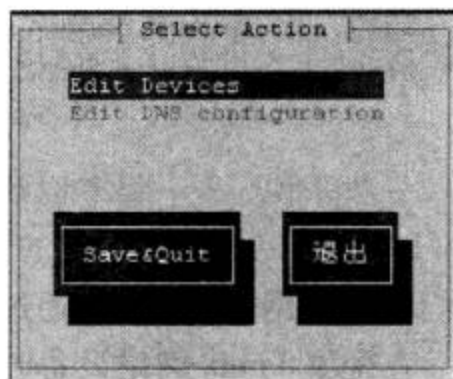


图 1-23 网卡和 DNS 配置界面

机不想使用 Internet 上的域名解析, 可以更改 hosts 文件, 加入自己的域名解析。

目前/etc/hosts 多用于集群环境或开发测试环境 (以免重新架构内网 DNS 服务器)。

4. 配置 DNS 域名解析服务器

配置 DNS 域名就比较简单了, 只需要配置/etc/resolv.conf 文件即可, 如下所示:

```
vim /etc/resolv.conf
nameserver 202.96.128.86
nameserver 202.96.128.166
```

resolv.conf 中最重要的选项是 nameserver, 它给出了要使用的名字服务器的 IP 地址。如果你通过 nameserver 选项指定了几个名字服务器。那么它们会以给出的先后顺序来决定主从服务器, 如果主服务器上没有对应的域名, 系统会自动从 DNS 上寻找。因此, 你首先应该给出最可靠的服务器。目前, 它至多支持 3 个服务器名字。

1.3.2 查看 Linux 服务器的网络连接

查看 Linux 服务器的网络连接时, 可以用以下命令: ifconfig、ping、netstat 等。它们都是我们会经常用到的查看网络连接方面的命令, 很实用。下面将分别介绍一下这几个命令的用法。

1. ifconfig

ifconfig 命令是用来显示所有网络接口的详细情况的, 可以显示/设置 IP 地址、子网掩码、广播地址等, 由于它只是暂时生效, 所以我还是推荐大家使用前面所介绍的网卡文件来设置。此命令和 Windows 的 ipconfig 命令很相似, 用法也很简单, 大家可以用 man 来了解其详细语法。

用 ifconfig 显示服务器的所有网络接口配置如下:

```
[root@localhost ~]# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 00:14:22:1B:71:20
          inet addr:203.93.236.146  Bcast:203.93.236.191  Mask:255.255.255.192
          inet6 addr: fe80::214:22ff:fe1b:7120/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:7864481 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8121233 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:1689037931 (1.5 GiB)  TX bytes:9937152687 (9.2 GiB)

eth1      Link encap:Ethernet  HWaddr 00:14:22:1B:71:21
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:48909 errors:0 dropped:0 overruns:0 frame:0
          TX packets:48909 errors:0 dropped:0 overruns:0 carrier:0
```

```

collisions:0 txqueuelen:0
RX bytes:27821974 (26.5 MiB) TX bytes:27821974 (26.5 MiB)

lo:0      Link encap:Local Loopback
inet addr:203.93.236.148 Mask:255.255.255.255
UP LOOPBACK RUNNING MTU:16436 Metric:1

sit0      Link encap:IPv6 - in - IPv4
NOARP MTU:1480 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:0 (0.0 b) TX bytes:0 (0.0 b)

```

如果只显示 eth0 的网络配置，则命令如下所示：

```

[root@localhost ~]# ifconfig eth0
eth0      Link encap:Ethernet HWaddr 00:14:22:1B:71:20
inet addr:203.93.236.146 Bcast:203.93.236.191 Mask:255.255.255.192
inet6 addr: fe80::214:22ff:fe1b:7120/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:7864647 errors:0 dropped:0 overruns:0 frame:0
TX packets:8121308 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:100
RX bytes:1689050791 (1.5 GiB) TX bytes:9937166253 (9.2 GiB)

```

如果想显示 eth0 的 IP 地址，命令如下所示：

```

[root@localhost ~]# ifconfig eth0 | grep "inet addr" | awk -F[:" "] + '{print $4}'
203.93.236.146

```

这里稍微解释一下此条命令，它是条 awk 语句，会以空格和 “:” 为分隔符，然后打印出第 4 列。

2. ping

相信大家都很熟悉 ping 命令了，它用于检查网络上某台主机是否为活动状态或是否发生故障。它会利用 TCP/IP 协议族中的 ICMP 协议的 ECHO_REQUEST 数据包强制从特定的主机上返回响应。一般而言，如果我们发现某网站无法访问，首先想到的可能就是这个命令。因为 ping 命令在执行过程中同样会涉及路由、地址解析、网关等，所以如果 ping 不通，则表明网络不正常。用其他方法也只能得到这样的结果，所以我们把它作为检测网络状态的首选。但是，由于它只经过底层的几个协议，因此即使它运行成功了，还是需要用其他的方法测试高层是否能提供所需的服务。

下面我们试着用 5 个数据包去 ping 一下 www.163.com：

```

ping -c 5 www.163.com
[root@localhost ~]# ping -c 5 www.163.com
PING 163.xdwscale.glb0.lxdns.com (210.51.213.180) 56(84) bytes of data.
64 bytes from 210.51.213.180: icmp_seq=1 ttl=62 time=1.28 ms
64 bytes from 210.51.213.180: icmp_seq=2 ttl=62 time=1.41 ms
64 bytes from 210.51.213.180: icmp_seq=3 ttl=62 time=1.38 ms
64 bytes from 210.51.213.180: icmp_seq=4 ttl=62 time=1.59 ms

```

```
64 bytes from 210.51.213.180: icmp_seq=5 ttl=62 time=1.31 ms
```

```
--- 163.xdwscache.glb0.lxdns.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3999ms
rtt min/avg/max/mdev = 1.286/1.399/1.598/0.117 ms
```

通常大家比较关心 ping 通的时间（它可以反映你与对方网站之间的连接速度）和有无丢包，这个命令的语法比较简单，大家可以多用 man 去了解其详细语法。

3. netstat

netstat 命令的功能是显示网络连接、路由表和网络接口的信息，可以让用户得知目前都有哪些网络连接正在运作。

下面是它的重要参数，以及详细的说明文字。

- -A：显示任何关联的协议控制块的地址。主要用于调试。
- -a：显示所有套接字的状态。在一般情况下不显示与服务器进程相关联的套接字。
- -i：显示自动配置接口的状态。那些在系统初始引导后配置的接口状态不在输出之列。
- -m：打印网络存储器的使用情况。
- -n：打印实际地址，而不是对地址的解释或显示主机、网络名之类的符号。
- -r：打印路由选择表。
- -f address：family 会对于给出名字的地址簇打印统计数字和控制块信息。到目前为止，它唯一支持的地址簇是 inet。
- -I interface：表示只打印给出名字的接口状态。
- -p protocol-name：表示只打印给出名字的协议的统计数字和协议控制块信息。
- -s：打印每个协议的统计数字。
- -t：表示在输出显示中用时间信息代替队列长度信息。

我们用得最多的也最习惯的有两个参数，即 netstat -an，如下所示：

```
[root@localhost ~]# netstat -an | grep -v unix
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address          State
tcp        0      0 127.0.0.1:2208          0.0.0.0:*                LISTEN
tcp        0      0 0.0.0.0:740             0.0.0.0:*                LISTEN
tcp        0      0 0.0.0.0:3306            0.0.0.0:*                LISTEN
tcp        0      0 0.0.0.0:111             0.0.0.0:*                LISTEN
tcp        0      0 0.0.0.0:80              0.0.0.0:*                LISTEN
tcp        0      0 127.0.0.1:631           0.0.0.0:*                LISTEN
tcp        0      0 127.0.0.1:25            0.0.0.0:*                LISTEN
tcp        0      0 127.0.0.1:6010          0.0.0.0:*                LISTEN
tcp        0      0 127.0.0.1:2207          0.0.0.0:*                LISTEN
tcp        0      0 203.93.236.146:80       117.34.91.54:4991       TIME_WAIT
tcp        0      0 203.93.236.146:80       117.34.91.54:1066       TIME_WAIT
tcp        0      0 203.93.236.146:80       117.34.91.54:3067       TIME_WAIT
tcp        0      0 :::ffff:127.0.0.1:8005  :::*                    LISTEN
tcp        0      0 :::8009                 :::*                    LISTEN
tcp        0      0 :::8080                 :::*                    LISTEN
tcp        0      0 :::22                   :::*                    LISTEN
```

```

tcp      0      0 0::1:6010          :::*               LISTEN
tcp      1      0 0::ffff:203.93.236.146:41844 ::ffff:203.93.236.146:3306 CLOSE_WAIT
tcp      1      0 0::ffff:203.93.236.146:42287 ::ffff:203.93.236.146:3306 CLOSE_WAIT
tcp      1      0 0::ffff:203.93.236.146:42289 ::ffff:203.93.236.146:3306 CLOSE_WAIT
tcp      0      52 0::ffff:203.93.236.146:22 ::ffff:220.249.72.138:23527 ESTABLISHED
udp      0      0 0.0.0.0:58153      0.0.0.0:*
udp      0      0 0.0.0.0:734        0.0.0.0:*
udp      0      0 0.0.0.0:737        0.0.0.0:*
udp      0      0 0.0.0.0:5353       0.0.0.0:*
udp      0      0 0.0.0.0:111        0.0.0.0:*
udp      0      0 0.0.0.0:631        0.0.0.0:*
udp      0      0 0:::33835          :::*
udp      0      0 0:::5353            :::*

```

Active UNIX domain sockets (servers and established)

Proto RefCnt Flags Type State I-Node Path

netstat -an 参数中 state 的含义如下所示。

- LISTEN: 侦听来自远方的 TCP 端口的连接请求。
- SYN-SENT: 在发送连接请求后等待匹配的连接请求。
- SYN-RECEIVED: 在收到和发送一个连接请求后等待对方对连接请求的确认。
- ESTABLISHED: 代表一个打开的连接, 我们常用此作为并发连接数。
- FIN-WAIT-1: 等待远程 TCP 连接中断请求, 或先前的连接中断请求的确认。
- FIN-WAIT-2: 从远程 TCP 等待连接中断请求。
- CLOSE-WAIT: 等待从本地用户发来的连接中断请求。
- CLOSING: 等待远程 TCP 对连接中断的确认。
- LAST-ACK: 等待原来发向远程 TCP 的连接中断请求的确认。
- TIME-WAIT: 等待足够的时间以确保远程 TCP 接收到连接中断请求的确认。
- CLOSED: 没有任何连接状态。

这里跟大家介绍一个我们经常在工作中用到的 shell 命令组合, 它是用来查看服务器网络连接状态并汇总的, 命令如下:

```

[root@localhost ~]# netstat -an | awk '/^tcp/ {++S[$NF]} END {for(a in S) print a, S[a]}'
TIME_WAIT 21
ESTABLISHED 185
LISTEN 9

```

参数说明:

- CLOSED: 没有连接是活动的或正在进行的。
- LISTEN: 服务器在等待进入呼叫。
- SYN_RECV: 一个连接请求已经到达, 等待确认。
- SYN_SENT: 应用已经开始, 打开一个连接。
- ESTABLISHED: 正常数据传输状态。它的值也可以近似理解为当前服务器的并发数。
- FIN_WAIT1: 应用说它已经完成。
- FIN_WAIT2: 另一边已同意释放。
- ITMED_WAIT: 等待所有分组死掉。

- CLOSING: 两边同时尝试关闭。
- TIME_WAIT: 另一边已初始化一个释放。
- LAST_ACK: 等待所有分组死掉。

很多时候我们需要查看系统的路由表,这时我们可以通过命令来查看。下面介绍在 Linux 下查看路由的方法,当然了我们也可以通过查看路由表来确定机器的网关,它的方法有两种。

第一种方法如下所示:

```
[root@localhost ~]# route -n
Kernel IP routing table
Destination      Gateway          Genmask          Flags  Metric  Ref    Use Iface
203.93.236.148   0.0.0.0         255.255.255.255  UH     0        0      0 lo
203.93.236.128   0.0.0.0         255.255.255.192  U      0        0      0 eth0
169.254.0.0      0.0.0.0         255.255.0.0      U      0        0      0 eth0
0.0.0.0         203.93.236.129 0.0.0.0         UG     0        0      0 eth0
```

所显示的内容中有 UG 的这行即是系统的默认网关。

第二种方法如下所示:

```
[root@localhost ~]# netstat -r
Kernel IP routing table
Destination      Gateway          Genmask          Flags  MSS  Window  irtt Iface
203.93.236.148   *               255.255.255.255  UH     0    0        0 lo
203.93.236.128   *               255.255.255.192  U      0    0        0 eth0
169.254.0.0      *               255.255.0.0      U      0    0        0 eth0
default         203.93.236.129 0.0.0.0         UG     0    0        0 eth0
```

在第二种方法中,我们用 traceroute 来随意跟踪一个网络地址(如下所示),其中第一条地址即是我们机器的网关,这种方法在 FreeBSD8.0 下经常用到。

```
[root@localhost ~]# traceroute www.163.com
traceroute to www.163.com (210.51.213.180), 30 hops max, 40 byte packets
1 203.93.236.129 (203.93.236.129) 0.861 ms 0.840 ms 0.825 ms
2 210.5.142.225 (210.5.142.225) 0.789 ms 0.779 ms 0.759 ms
3 218.106.127.82 (218.106.127.82) 1.478 ms 1.468 ms 1.456 ms
4 218.104.110.82 (218.104.110.82) 1.427 ms 1.429 ms 1.651 ms
5 220.249.83.130 (220.249.83.130) 1.639 ms 2.114 ms 2.349 ms
6 210.51.213.180 (210.51.213.180) 1.585 ms 1.670 ms 1.432 ms
```

很多时候,我们需要追踪网络数据包的路径,这个时候我们该用什么命令呢?其实可以用 traceroute 来轻松解决这个问题。traceroute 命令的功能是追踪网络数据包的路由途径,数据包默认的大小为 40 字节。跟踪到目的地需要经过几个路由器,所以,简单来说这个命令就是用来进行路由跟踪的。命令的用法如下所示:

```
[root@localhost ~]# traceroute www.163.com
traceroute to www.163.com (210.51.213.180), 30 hops max, 40 byte packets
1 203.93.236.129 (203.93.236.129) 0.861 ms 0.840 ms 0.825 ms
2 210.5.142.225 (210.5.142.225) 0.789 ms 0.779 ms 0.759 ms
3 218.106.127.82 (218.106.127.82) 1.478 ms 1.468 ms 1.456 ms
4 218.104.110.82 (218.104.110.82) 1.427 ms 1.429 ms 1.651 ms
```

```

5 220.249.83.130 (220.249.83.130) 1.639 ms 2.114 ms 2.349 ms
6 210.51.213.180 (210.51.213.180) 1.585 ms 1.670 ms 1.432 ms

```

以上命令显示了我们的机器到达 `www.163.com` 的数据包之间的完整路由，1 表示离我们最近的路由器的 IP 地址，其他以此类推。

4. nslookup

`nslookup` 命令的功能是查询一台机器的 IP 地址和与其对应的域名。通常需要一台域名服务器来提供域名服务，如果用户已经设置好域名服务器，就可以用这个命令来查看不同主机的 IP 地址所对应的域名。它的用法也很简单，如下所示：

```

[root@localhost ~]# nslookup
> mail.163.com
Server:      218.104.111.114
Address:     218.104.111.114#53

Non-authoritative answer:
mail.163.com canonical name = mcache.mail.163.com.
mcache.mail.163.com canonical name = email.163.com.lxdns.com.
email.163.com.lxdns.com canonical name = 163.xxcache.z.lxdns.com.
163.xxcache.z.lxdns.com canonical name = 06811.xdwscache.glb0.lxdns.com.
Name: 06811.xdwscache.glb0.lxdns.com
Address: 210.51.213.180
> exit

```

5. dig

如果大家不喜欢 `nslookup` 命令的这种交互式用法，我们可以用 `dig` 命令来查看。值得一说的是，`nslookup` 存在于 Windows 系列和 Linux 及 FreeBSD 等系统中，它比较常用；而 `dig` 只存在于 Linux 及 FreeBSD 等开源系统下。`dig` 最基本的用法如下所示：

```

dig sina.com.cn                //查询 A 记录
;; QUESTION SECTION:
;sina.com.cn.                  IN      A
;; ANSWER SECTION:
sina.com.cn.                  37      IN      A      202.108.33.32

dig sina.com.cn ns              //查询 NS 记录
;; QUESTION SECTION:
;sina.com.cn.                  IN      NS
;; ANSWER SECTION:
sina.com.cn.                  21478   IN      NS      ns2.sina.com.cn.
sina.com.cn.                  21478   IN      NS      ns3.sina.com.cn.
sina.com.cn.                  21478   IN      NS      ns1.sina.com.cn.

dig sina.com.cn soa             //查询 SOA 记录
;; QUESTION SECTION:
;sina.com.cn.                  IN      SOA
;; ANSWER SECTION:
sina.com.cn.                  600     IN      SOA      ns1.sina.com.cn. senjin.staff.sina.com.cn. 5 1800
600 604801 600

```

另外, 还可以通过 `dig @ Server sina.com.cn.` 在 Server 服务器上查询 `sina.com.cn` 的记录。比如:

```
dig @210.51.191.22 sina.com.cn
;; QUESTION SECTION:
;sina.com.cn.                IN      A

;; AUTHORITY SECTION:
sina.com.cn.                 21600 IN  NS     ns1.sina.com.cn.
sina.com.cn.                 21600 IN  NS     ns2.sina.com.cn.
sina.com.cn.                 21600 IN  NS     ns3.sina.com.cn.

;; ADDITIONAL SECTION:
ns1.sina.com.cn.             21600 IN  A       202.106.184.166
ns2.sina.com.cn.             21600 IN  A       61.172.201.254
ns3.sina.com.cn.             21600 IN  A       202.108.44.55
```

从根服务器开始追踪一个域名的解析过程, 可以用如下命令:

```
[root@localhost ~]# dig www.163.com +trace

; << >> DiG 9.3.6 - Pl - RedHat - 9.3.6 - 4.Pl.el5_4.2 << >> www.163.com +trace
;; global options: printcmd
.                518380 IN  NS  l.root-servers.net.
.                518380 IN  NS  a.root-servers.net.
.                518380 IN  NS  g.root-servers.net.
.                518380 IN  NS  e.root-servers.net.
.                518380 IN  NS  c.root-servers.net.
.                518380 IN  NS  b.root-servers.net.
.                518380 IN  NS  i.root-servers.net.
.                518380 IN  NS  d.root-servers.net.
.                518380 IN  NS  h.root-servers.net.
.                518380 IN  NS  m.root-servers.net.
.                518380 IN  NS  f.root-servers.net.
.                518380 IN  NS  k.root-servers.net.
.                518380 IN  NS  j.root-servers.net.
;; Received 500 bytes from 218.104.111.114#53 (218.104.111.114) in 2 ms

com.             172800 IN  NS  m.gtld-servers.net.
com.             172800 IN  NS  d.gtld-servers.net.
com.             172800 IN  NS  i.gtld-servers.net.
com.             172800 IN  NS  e.gtld-servers.net.
com.             172800 IN  NS  l.gtld-servers.net.
com.             172800 IN  NS  c.gtld-servers.net.
com.             172800 IN  NS  h.gtld-servers.net.
com.             172800 IN  NS  j.gtld-servers.net.
com.             172800 IN  NS  f.gtld-servers.net.
com.             172800 IN  NS  k.gtld-servers.net.
com.             172800 IN  NS  g.gtld-servers.net.
com.             172800 IN  NS  a.gtld-servers.net.
com.             172800 IN  NS  b.gtld-servers.net.
;; Received 501 bytes from 198.41.0.4#53 (a.root-servers.net) in 256 ms
```

```
163.com.          172800 IN NS ns2.nease.net.
163.com.          172800 IN NS ns3.nease.net.
163.com.          172800 IN NS ns4.nease.net.
;; Received 140 bytes from 192.55.83.30#53 (m.gtld-servers.net) in 56 ms
```

```
www.163.com.      600 IN CNAME www.cache.wangsu.netease.com.
www.cache.wangsu.netease.com. 600 IN CNAME www.163.com.lxdns.com.
;; Received 100 bytes from 114.113.197.12#53 (ns2.nease.net) in 49 ms
```

6. finger

`finger` 命令的功能是查询用户的信息，通常会显示系统中某个用户的用户名、主目录、停滞时间、登录时间、登录 shell 等信息。W 的命令效果与其类似。其用法如下所示：

```
[root@localhost ~]# finger
Login      Name      Tty      Idle   Login Time   Office      Office Phone
root       root      pts/2                Mar 29 09:02 (220.249.72.138)

lsuf
```

7. lsof

`lsof` (list open files) 是一个列出当前系统打开文件的工具。在 Unix 环境下，任何事物都是以文件的形式存在的，通过文件不仅仅可以访问常规数据，还可以访问网络连接和硬件。像传输控制协议 (TCP) 和用户数据报协议 (UDP) 套接字等，系统在后台都为应用程序分配了一个文件描述符，无论这个文件的本质如何，该文件描述符都会为应用程序与基础操作系统之间的交互提供通用接口。因为应用程序打开文件的描述符列表提供了大量关于这个应用程序的信息，因此通过 `lsof` 工具查看这个列表对系统监测以及排错非常有帮助。顺便提一下，这个工具首先出现在 Unix 系统中，后来才移植到 Linux 平台下。平时我们用得最多的是 `lsof -i`，用来查看特定端口的情况，比如，可以用 `lsof -i:22` 查看 22 端口是由哪些程序占用的，如下所示：

```
[root@localhost ~]# lsof -i:22
COMMAND  PID USER  FD  TYPE  DEVICE SIZE NODE NAME
sshd     2934 root   3u   IPv6   8903      TCP *:ssh (LISTEN)
sshd     20573 root   3u   IPv6 2158289      TCP 203.93.236.146:ssh->220.249.72.138:23527 (ESTABLISHED)
```

8. Socketstat

在 FreeBSD8 或 FreeBSD8.1 中，我们可以使用 `sockstat` 查看打开的套接字的情况，包括端口相应进程的进程名、PID、用户等。用得最多的是 `sockstat -4l`（它非常好用，也可以用在 Jail 虚拟机上）。另外，说明一下，此命令只能用于 FreeBSD 等 Unix 系统中，Centos5.5 中无此命令。

Socket 的参数说明如下所示：

```
NAME
sockstat -- list open sockets
```

```
SYNOPSIS
sockstat [-46clu] [-p ports]

DESCRIPTION
```


The sockstat command lists open Internet or UNIX domain sockets. The following options are available:

- 4 Show AF_INET (IPv4) sockets.
- 6 Show AF_INET6 (IPv6) sockets.
- c Show connected sockets.
- l Show listening sockets

1.3.3 查看 Linux 服务器的进程

Linux 服务器正常启动后, 提供服务时会调用程序, 占用进程。这时候我们如何查看系统中有哪些进程在被调用呢? 我们可以通过以下命令来查看。

1. ps

ps 命令是最基本同时也是非常强大的进程查看命令。使用该命令可以确定有哪些进程正在运行和它所运行的状态、进程是否结束、进程有没有僵死、哪些进程占用了过多的资源等。总之大部分信息都是可以通过执行该命令得到的。ps 命令最常用来监控后台进程的工作情况, 因为后台进程是不和屏幕、键盘这些标准输入/输出设备进行通信的, 所以如果需要检测后台情况, 就需要使用 ps 命令了。它的格式如下所示:

格式: ps [选项]

其主要选项如下。

- -a: 显示系统中所有进程的信息。
- -e: 显示所有进程的信息。
- -f: 显示进程的所有信息。
- -l: 以长格式显示进程信息。
- -r: 只显示正在运行的进程。
- -u: 显示面向用户的格式 (包括用户名、CPU 及内存的使用情况等信息)。
- -x: 显示所有非控制终端上的进程信息。
- -p: 显示由进程 ID 指定的进程信息。
- -t: 显示指定终端上的进程信息。

要对进程进行监测和控制, 首先要了解当前进程的情况, 当然也就需要查看当前进程的状态了。通过 ps 命令查看进程, 根据所显示的信息可以了解哪个进程正在运行、哪个进程被挂起了、进程已运行了多久、进程正在使用的资源、进程的相对优先级, 以及进程的标志号 (PID)。所有这些对用户都很有用, 对于系统管理员来说尤为重要。使用 ps -aux 命令可以获得终端上所有用户的有关进程的所有信息, 这个也是我们平时用得最多的命令之一, 如下所示:

```
[root@localhost ~]# ps -aux
Warning: bad syntax, perhaps a bogus '- '? See /usr/share/doc/procps-3.2.7/FAQ
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0 10348  688 ?        Ss   Mar16   0:01 init [5]
root         2  0.0  0.0   0      0 ?        S<   Mar16   0:00 [migration/0]
root         3  0.0  0.0   0      0 ?        SN   Mar16   0:00 [ksoftirqd/0]
root         4  0.0  0.0   0      0 ?        S<   Mar16   0:00 [watchdog/0]
root         5  0.0  0.0   0      0 ?        S<   Mar16   0:00 [migration/1]
root         6  0.0  0.0   0      0 ?        SN   Mar16   0:00 [ksoftirqd/1]
...
```

```

avahi      3153  0.0  0.0   23148   336 ?      Ss  Mar16  0:00 avahi - daemon: chroot helper
root       3214  0.0  0.0   18416   476 ?      S   Mar16  0:00 /usr/sbin/smartd -q never
root       3219  0.0  0.0    3792   488 tty1    Ss + Mar16  0:00 /sbin/mingetty tty1
root       3220  0.0  0.0    3792   484 tty2    Ss + Mar16  0:00 /sbin/mingetty tty2
root       3221  0.0  0.0    3792   488 tty3    Ss + Mar16  0:00 /sbin/mingetty tty3
root       3222  0.0  0.0    3792   488 tty4    Ss + Mar16  0:00 /sbin/mingetty tty4
root       3224  0.0  0.0    3792   488 tty5    Ss + Mar16  0:00 /sbin/mingetty tty5
root       3226  0.0  0.0    3792   488 tty6    Ss + Mar16  0:00 /sbin/mingetty tty6
root       3233  0.0  0.0  169912  2756 ?      Ss  Mar16  0:00 /usr/sbin/gdm-binary -nodaemon
root       3322  0.0  0.0  197124  2564 ?      S   Mar16  0:00 /usr/sbin/gdm-binary -nodaemon
root       3324  0.0  0.1  189808  4112 ?      Sl  Mar16  0:00 /usr/libexec/gdm-rh-security
-token-helper
root       3325  0.0  0.1   90548  6264 tty7    Ss + Mar16  0:02 /usr/bin/Xorg :0 -br -audit 0
-auth /var/gdm/:0.Xauth -noli
root       3339  0.0  0.4  257920 16784 ?      SN  Mar16  0:20 /usr/bin/python -tt /usr/sbin/
yum-updatesd
root       3341  0.0  0.0   12916   1188 ?      SN  Mar16  0:00 /usr/libexec/gam_server
gdm        3351  0.0  0.8  379612 36176 ?      Ss  Mar16  0:00 /usr/libexec/gdmgreeter
root       4342  0.0  0.0   41096    896 ?      Ss  Mar16  0:00 nginx: master process /usr/local/
webserver/nginx/sbin/nginx
www        4343  0.0  0.6   65920 26232 ?      S   Mar16  0:05 nginx: worker process
www        4344  0.0  0.6   65920 26160 ?      S   Mar16  0:02 nginx: worker process
www        4345  0.0  0.6   66076 26460 ?      S   Mar16  0:03 nginx: worker process
www        4346  0.0  0.6   65920 26104 ?      S   Mar16  0:03 nginx: worker process
www        4347  0.0  0.6   66052 26228 ?      S   Mar16  0:04 nginx: worker process
www        4348  0.0  0.6   66012 26372 ?      S   Mar16  0:04 nginx: worker process
www        4349  0.0  0.6   65788 26076 ?      S   Mar16  0:03 nginx: worker process
www        4350  0.0  0.6   65920 26188 ?      S   Mar16  0:06 nginx: worker process
root       5314  0.0  0.0     0     0 ?      S   Mar28  0:00 [pdflush]
root       5315  0.0  0.0     0     0 ?      S   Mar28  0:00 [pdflush]
root       6767  0.0  0.0   68284  1564 tty8    Ss + Mar17  0:00 /bin/bash
root      10369  0.0  0.0   65556   932 pts/2    R+  14:23  0:00 ps -axu
root      18906  0.0  6.4  1443024 261272 ?      Sl  00:15  0:46 /usr/local/jdk/bin/java -Dja-
va.util.logging.config.file=/us
root      20573  0.0  0.0   90140   3344 ?      Ss  09:02  0:00 sshd: root@pts/2
root      20575  0.0  0.0   68412  1756 pts/2    Ss  09:02  0:00 -bash

```

□ USER: 表示启动进程的用户。

□ PID: 表示进程标志号。

□ % CPU: 表示运行该进程占用 CPU 的时间与该进程总的运行时间之比。

□ % MEM: 表示该进程占用内存与总内存之比。

□ VSZ: 表示占用的虚拟内存大小, 以 KB 为单位。

□ RSS: 为进程占用的物理内存值, 以 KB 为单位。

□ TTY: 表示该进程建立时所对应的终端, “?” 表示该进程不占用终端。

□ STAT: 表示进程的运行状态。包括以下几种代码: D, 不可中断的睡眠; R, 就绪 (在可运行队列中); S, 睡眠; T, 被跟踪或停止; Z, 终止 (僵死) 的进程, 这些进程不存在, 但暂时无法消除; W, 没有足够的内存分页可分配; <, 高优先级的进程; N, 低优先级的进

程；L，有内存分页分配并锁在内存体内（实时系统或 I/O）。

□ START：为进程开始时间。

□ TIME：为执行的时间。

□ COMMAND：是对应的命令名。

由于 ps 执行后结果太多了，所以我们一般会带上 grep 参数来精确定位我们需要的进程号。例如，我们要查看 Nginx 占用的进程，可以用命令 ps -axu | grep nginx，如下所示：

```
[root@localhost ~]# ps -axu | grep -v grep | grep nginx
Warning: bad syntax, perhaps a bogus '- '? See /usr/share/doc/procps-3.2.7/FAQ
root      4342  0.0  0.0  41096   896 ?        Ss   Mar16   0:00 nginx: master process /usr/local/webserver/nginx/sbin/nginx
www      4343  0.0  0.6  65920 26232 ?        S    Mar16   0:05 nginx: worker process
www      4344  0.0  0.6  65920 26160 ?        S    Mar16   0:02 nginx: worker process
www      4345  0.0  0.6  66076 26460 ?        S    Mar16   0:03 nginx: worker process
www      4346  0.0  0.6  65920 26104 ?        S    Mar16   0:03 nginx: worker process
www      4347  0.0  0.6  66052 26228 ?        S    Mar16   0:04 nginx: worker process
www      4348  0.0  0.6  66012 26372 ?        S    Mar16   0:04 nginx: worker process
www      4349  0.0  0.6  65788 26076 ?        S    Mar16   0:03 nginx: worker process
www      4350  0.0  0.6  65920 26188 ?        S    Mar16   0:06 nginx: worker process
```

2. top

top 命令可动态显示服务器的进程信息。top 命令和 ps 命令的基本作用是相同的，都显示系统当前进程的状况。但 top 是一个动态显示过程，即用户可以通过按键来不断刷新当前状态。此命令的使用举例如下所示：

```
top - 22:26:54 up 5 days, 22:39,  4 users,  load average: 0.69, 0.57, 0.44t
Tasks: 162 total,  1 running, 160 sleeping,  0 stopped,  1 zombie
Cpu(s):  1.0%us,  0.3%sy,  0.0%ni, 85.9%id,  0.0%wa,  0.3%hi, 12.5%si,  0.0%st
Mem:   8166152k total,  4328628k used,  3837524k free,  245328k buffers
Swap: 10482404k total,    0k used, 10482404k free,  2689332k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
16646	glenx	17	0	2643m	1.0g	11m	S	7.0	12.7	19:54.24	java
20610	root	15	0	12740	1124	808	R	0.3	0.0	0:00.08	top
1	root	15	0	10348	636	540	S	0.0	0.0	0:00.59	init
2	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	migration/0
3	root	34	19	0	0	0	S	0.0	0.0	0:00.01	ksoftirqd/0
4	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	watchdog/0
5	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	migration/1
6	root	34	19	0	0	0	S	0.0	0.0	0:00.00	ksoftirqd/1
7	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	watchdog/1
8	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	migration/2
9	root	34	19	0	0	0	S	0.0	0.0	0:00.00	ksoftirqd/2
10	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	watchdog/2
11	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	migration/3
12	root	34	19	0	0	0	S	0.0	0.0	0:00.00	ksoftirqd/3
13	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	watchdog/3
14	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	events/0


```

15 root      10   -5     0     0     0 S  0.0  0.0   0:00.00 events/1
16 root      10   -5     0     0     0 S  0.0  0.0   0:00.00 events/2
17 root      10   -5     0     0     0 S  0.0  0.0   0:00.00 events/3

```

第一行内容依次表示当前时间、系统启动的时间、当前系统登录的用户数、平均负载。第二行依次显示的是所有启动的、目前运行的、挂起（Sleeping）的和无用（Zombie）的进程。第三行显示的是目前 CPU 的使用情况，包括系统占用的比例、用户使用比例、闲置（Idle）比例。第四行显示物理内存的使用情况，包括总的可以使用的内存、已用内存、空闲内存、缓冲区占用的内存。第五行显示交换分区的使用情况，包括总的交换分区、使用的、空闲的和用于高速缓存的交换分区。第六行显示的内容最多，下面分别进行详细解释。

- PID（Process ID）：进程标志号，是非零正整数。
- USER：进程所有者的用户名。
- PR：进程的优先级别。
- NI：进程的优先级别数值。
- VIRT：进程占用的虚拟内存值。
- RES：进程占用的物理内存值。
- SHR：进程使用的共享内存值。
- STAT：进程的状态，其中 S 表示休眠，R 表示正在运行，Z 表示僵死状态，N 表示该进程优先值是负数。
- %CPU：该进程占用的 CPU 使用率。
- %MEM：该进程占用的物理内存和总内存的百分比。
- TIME：该进程启动后占用的总 CPU 时间。
- COMMAND：进程启动的启动命令名称，如果这一行显示不下，在进程中会有一个完整的命令行。

在 top 命令使用过程中，还可以使用一些交互的命令来完成其他参数的功能。这些命令是通过快捷键启动的，格式如下所示：

<空格>：立刻刷新

其主要参数如下所示。

- P：根据 CPU 使用的多少进行排序。
- T：根据时间、累计时间排序。
- q：退出 top 命令。
- m：切换显示内存信息。
- t：切换显示进程和 CPU 状态信息。
- c：切换显示命令名称和完整命令行。
- M：根据使用内存的大小进行排序。
- W：将当前设置写入 ~/.toprc 文件中，这是写 top 配置文件的推荐方法。

3. pgrep 命令

pgrep 命令的作用是查找当前运行的进程，并列出匹配给定条件进程的 PID。所有的条件都必须匹配才会被列出。使用权限为所有用户。

以下是我在一台 64 位 FreeBSD8.1 x86_64 的机器上以用户名 andrewyu 查看的 Nginx 的所有进程号。

```
%pgrep nginx
2834
2833
2832
2831
2830
2828
2827
2826
2825
```

4. kill 命令

kill 命令的作用是终止一个进程。其格式如下：

```
kill [-s signal | -p] [-a] pid...
kill -l [ signal ]
```

它的主要选项如下所示。

- -s: 指定发送的信号。
- -p: 模拟发送信号。
- -l: 指定信号的名称列表。
- pid: 要终止的进程的 ID 号。
- signal: 表示信号。

kill 可将指定的信息送至程序中。预设的信息为 SIGTERM (15)，可将指定程序终止。若仍无法终止该程序，则可以使用 SIGKILL (9) 信息尝试强制删除程序。kill 命令的工作原理是，向 Linux 系统的内核发送一个系统操作信号和某个程序的进程标志号，然后系统内核就可以对进程标志号指定的进程进行操作了。当需要中断一个前台进程的时候，通常使用 Ctrl + C 组合键；但是对于一个后台进程来说，就不是一个组合键所能解决的了，这时就必须使用 kill 命令。另外，kill -9 可以强制杀掉进程，尤其适用于僵尸进程。

5. killall

killall 命令的作用是通过程序的名字，直接杀死所有进程，这里简单介绍一下。如果要杀掉 nginx 的所有进程，要是用 kill 的话就要执行 9 次，但如果用 killall nginx 则能很轻松地一次性解决问题。另外，大家有兴趣也可以了解一下 pkill 的用法，在服务器里用它来“踢人”还是很方便的，这个命令在工作中我们用得不多，大家稍微了解一下即可。

在 Linux 下，最强大的进程管理命令莫过于 ps 和 top 了，我们应该掌握它们的详细语法，在工作中灵活地使用它们。

1.3.4 在 Centos5.5、FreeBSD8.1 及 Windows 下添加静态路由

目前公司所用的服务器系统种类繁多，有 FreeBSD8.1 x86_64、64bit Windows 2003 R2 和 Centos5.5 x86_64 等，并且所规划的网段也不少，比如 192.168.4.0、192.168.10.0、192.168.20.0、192.168.21.0，还有 10.1.0.0 等，其中 192.168.4.0 属于办公网络；192.168.20.0 和 192.168.21.0 属于开发环境，192.168.10.0 和 10.1.0.0 属于线上环境（对外提供服务）。很多时候公司的路由器

及防火墙都做了严格控制（因权限方面的原因不能随便更改其规则），这时就需要手动在服务器或客户机上添加路由了。本节将简单归纳一下如何手动添加路由，希望对大家的工作有所帮助。

先简要说一下最简单的 Windows 2003 和 Windows XP、Windows 7 下的静态路由配置，如下所示：

```
route add 192.168.4.0 mask 255.255.255.0 192.168.4.2
```

`route add` 是 Windows 下 `route` 添加网段的特定语法（注意，非 `route add -net`），192.168.4.0 是需要路由的网段，`mask` 后面是此网段的子网掩码，192.168.4.2 是下一条地址，这里要注意用 `route` 操作时跟 Centos 和 FreeBSD 的区别，不然很容易混淆。

如果要永久添加路由该怎么做呢？

`route` 命令可以在 Windows 2000、Windows 2003、Windows XP 和 Windows 7 等操作系统中手动配置静态路由，但是重启之后路由便丢失了，还要重新增加。那么如何让路由一直保持着，重启后仍然存在呢？其实很简单，在增加路由的命令后加上 `-p` 开关就可以了。例如：

```
route -p add 10.10.0.0 mask 255.255.0.0 192.168.4.222
```

参数说明：

`-p` 与 `add` 命令共同使用时，可指定路由被添加到注册表中，并在启动 TCP/IP 协议的时候初始化 IP 路由表。默认情况下，启动 TCP/IP 协议时不会保存添加的路由。`-p` 在与 `print` 命令一起使用时，则会显示永久路由列表。所有其他的命令都忽略此参数。永久路由存储在注册表中的位置在如下所示的完整路径里（如图 1-24 所示）。

HKEY_LOCAL_MACHINE\CurrentControlSet\Services\Tcpip\Parameters\PersistentRoutes

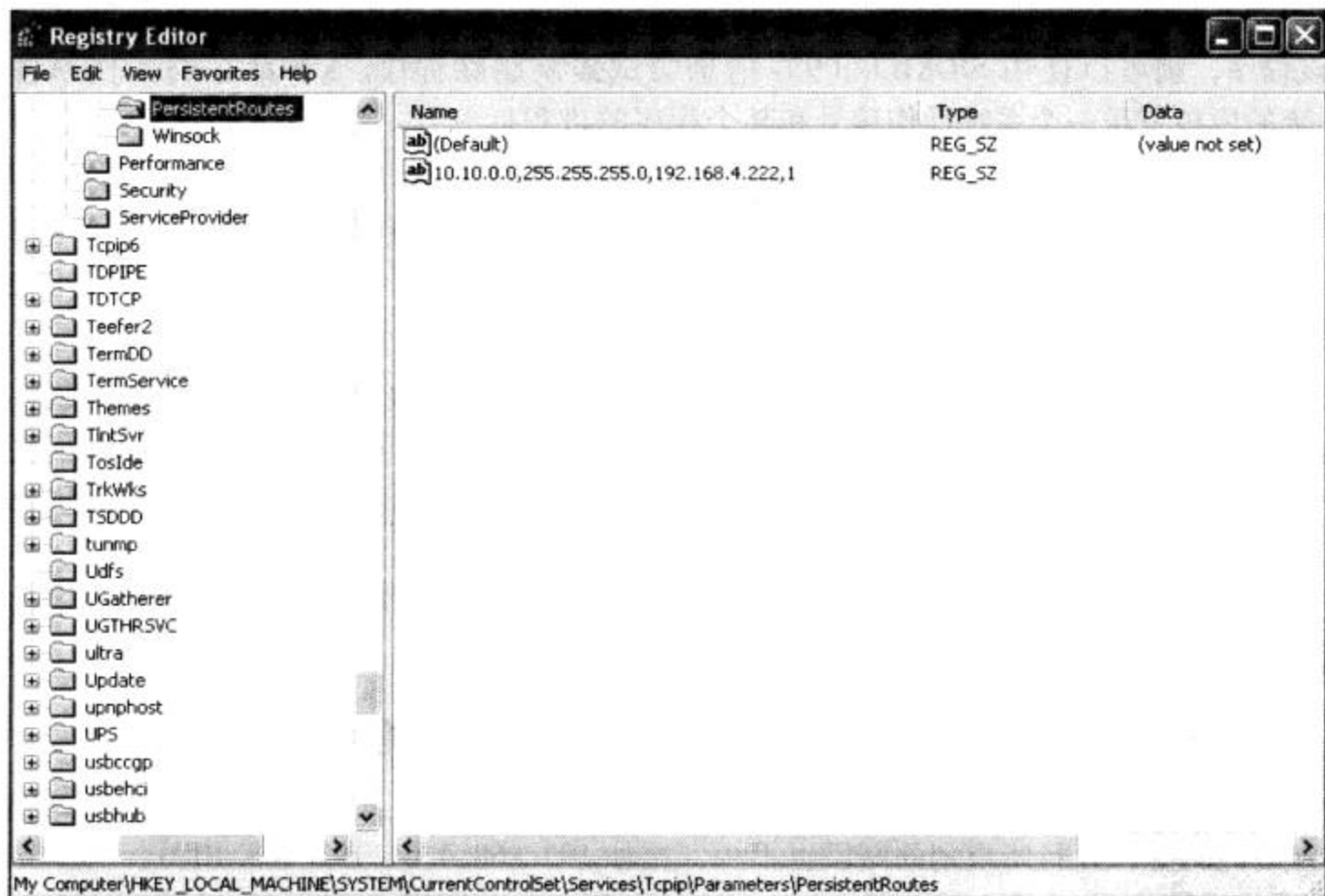


图 1-24 Windows 中注册表存放静态路由的位置

注意图 1-25 中“Persistent Routes”下面的内容，它表示无论重启或关机，此条静态路由在我们的机器上是永久生效的。

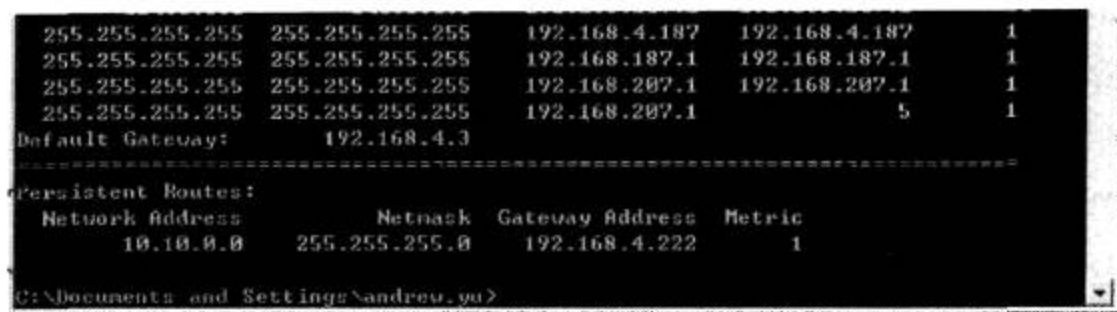


图 1-25 Persistent Routes 表示永久静态路由

下面再介绍一下 Centos5.5 中比较常见的永久添加静态路由的方法。手动添加路由的方式如下所示：

```
route add -net 172.16.6.0 netmask 255.255.255.0 gw 172.16.2.25
```

其中，route add -net 是 Centos 下添加静态路由网络的方式，netmask 是网络段的子网掩码，gw 表示下一条地址，其实就是指 172.16.6.0 网段的路由通过 172.16.2.25 出去。这只是临时的效果，如果服务器重启则失效，如何能让其永久生效呢？可以将其写进文件中，如下所示：

```
vim /etc/sysconfig/network-scripts/route-eth0
172.16.6.0/24 via 172.16.2.25
```

写进文件后，服务器重启后也不需要担心了，它会一直生效。在 Centos 下查看路由表的方式很多，netstat -rn 或 route -n 均可。Linux 中相关资料也比较多，这里就不重点说明了。

在 FreeBSD 中添加路由并不常见，相对于上面两种服务器而言算很少了，而且有时还很容易混淆。如果要添加一个网段为 172.16.6.0，下一个地址为 172.16.2.25 的路由，如下所示：

```
route add -net 172.16.0.0/24 172.16.2.25
```

注意它与 Centos5 和 Windows 2003 的区别，它后面不接子网掩码这个参数（即 172.16.0.0/24），如果硬要接上 mask 参数的话，shell 会产生报错信息 bad netmask。

如果要在 FreeBSD 中添加永久的路由，其实也很简单，即在 /etc/rc.conf 的最后添加相应的命令，如下命令行可以为你的 FreeBSD8 机器添加 4 条永久静态路由：

```
static_routes="net1 net2 net3 net4"
route_net1="-net 192.168.4.0/24 192.168.21.3"
route_net2="-net 192.168.10.0/24 192.168.21.3"
route_net3="-net 192.168.20.0/24 192.168.21.3"
route_net4="-net 10.1.0.0/16 192.168.21.3"
```

本节主要介绍了如何通过命令来配置和监控 Linux 服务器的网络设置及进程状态，由于网络对于 Linux 服务器来说意义重大，所以这一节的内容相对比较重要，建议大家熟练掌握。

1.4 Linux 服务器的日志管理

从安全的角度来说，Linux 服务器的日志非常重要，它记录了系统每天所发生的各种各样的事情，如果服务器受到攻击，就可以根据它来进行分析。同时，它还是很重要的排障依据，可以通过

它来检查错误发生的原因，所以我们必须了解和熟悉其运作机制。

1.4.1 系统日志 syslog.conf 的配置详解

目前，Linux 依旧使用 syslog 作为日志监控进程，因此对其进行必要的配置能减少很多麻烦，并可更有效地从系统日志监控到系统的状态。理解并完善一个 syslog 的配置，对于系统管理员来说尤为重要，/etc/syslog.conf 根据如下的格式定义规则：

```
facility.level action
```

即“设备.优先级 处理方案”。

facility.level 字段也称为 selector（选择条件），选择条件和处理方案之间用空格或 Tab 分隔开。#号开头的是注释，空白行会自动跳过。下面分别介绍。

(1) facility（设备）

facility 定义了日志消息的范围，可以使用的 key 如下所示。

- ☐ auth：由 pam_pwd 报告的认证活动。
- ☐ authpriv：包括特权信息，如用户名在内的认证活动。
- ☐ cron：与 cron 和 at 有关的计划任务信息。
- ☐ daemon：与 inetd 守护进程有关的后台进程信息。
- ☐ kern：内核信息，首先通过 klogd 传递。
- ☐ lpr：与打印服务有关的信息。
- ☐ mail：与电子邮件有关的信息。
- ☐ mark：syslog 的内部功能，用于生成时间戳。
- ☐ news：来自新闻服务器的信息。
- ☐ syslog：由 syslog 生成的信息。
- ☐ user：由用户程序生成的信息。
- ☐ uucp：由 uucp 生成的信息。
- ☐ local0 ~ local7：与自定义程序一起使用。

另外，* 通配符代表除 mark 以外的所有功能。security 是一个旧的 key 定义，等同于 auth，不建议使用。

(2) level（优先级）

level 定义了消息的紧急程度。按严重程度由高到低顺序排列如下。

- ☐ emerg：该系统不可用，等同于 panic。
- ☐ alert：需要立即被修改的条件。
- ☐ crit：危急情况。
- ☐ err：错误消息，等同于 error。
- ☐ warning：预警信息，等同于 warn。
- ☐ notice：具有重要性的普通条件。
- ☐ info：提供信息的消息。
- ☐ debug：不包含函数条件或问题的其他信息。
- ☐ none：没有重要级，通常用于排错。

另外，* 代表所有级别，除了 none 外，panic、error、warn 均为旧的标识符，不建议使用。

在定义 level 级别的时候，需要注意两点：

- 优先级是由应用程序在编程的时候决定的，除非修改源码再编译，否则不能改变消息的优先级。
- 低优先级包含高优先级，例如，为某个应用程序定义 info 的日志导向，则涵盖 notice、warning、err、crit、alert、emerg 等消息。

(3) selector (选择条件)

通过小数点符号“.”把 facility 和 level 连接在一起则成为 selector (选择条件)。可以使用分号“;”同时定义多个选择条件。也支持如下 3 个修饰符。

- *：表示所有日志信息。
- =：等于，即仅包含本优先级的日志信息。
- !：不等于，本优先级日志信息除外。

(4) action (处理方案)

由前面选择条件定义的日志信息，可执行下面的动作。

- file：指定日志文件的绝对路径。
- terminal 或 print：发送到串行或并行设备的标志符，例如/dev/ttyS2 @ host 表示远程的日志服务器。
- username：发送信息到本机的指定用户信息窗口中，但该用户必须已经登录到系统中。
- named pipe：发送到预先使用 mkfifo 命令创建的 FIFO 文件的绝对路径中。

注意 不能通过“|/var/xxx.sh”方式将日志导入到其他脚本中处理。

1.4.2 Linux 下的日志维护技巧

1. 系统日志

/var/log/messages 不仅是服务器的系统日志，很多时候它也包括许多服务的日志，所以它被称为“杂货铺”，建议重点关注。大家一般都喜欢用以下命令来看最后 10 条日志：tail -n10 /var/log/messages。

其实还可以将一段日志保存成文件（Xmanager3.0 企业版的 shell 也有日志录像截取功能），或者直接用 vim 来处理。我以前配置主从复制的 bind 服务器时，有时会因为权限的原因报错，这时就可以在一台报错的服务器上用命令 tail -f /var/log/messages 实时查看服务器的日志变化情况，从而查找错误的蛛丝马迹。事实证明，效果很好，而且将此命令用于 lvs + keepalived 的排错效果也不错。其他服务器配置排错以此类推，这个做法也推荐读者掌握。

2. 系统安全日志

/var/log/secure 记录登录系统存取数据的文件，例如 POP3、SSH、Telnet、FTP 等都会被记录，我们可以利用此文件找出不安全的登录 IP。目前比较流行的 SSH 防暴力破解工具 DenyHosts 主要也是读此文件。另外，我写了一个原理类似的 shell 安全脚本，用于线上服务器，在后面的章节跟大家分享。

operator	** 从未登录过**
games	** 从未登录过**
gopher	** 从未登录过**
ftp	** 从未登录过**
nobody	** 从未登录过**
nscd	** 从未登录过**
vcsa	** 从未登录过**
pcap	** 从未登录过**
rpc	** 从未登录过**
apache	** 从未登录过**
mailnull	** 从未登录过**
smmsp	** 从未登录过**
ntp	** 从未登录过**
hsqldb	** 从未登录过**
xfs	** 从未登录过**
rpcuser	** 从未登录过**
sshd	** 从未登录过**
dbus	** 从未登录过**
avahi	** 从未登录过**
haldaemon	** 从未登录过**
avahi - autoipd	** 从未登录过**
gdm	** 从未登录过**
longfei	** 从未登录过**
ldap	** 从未登录过**
www	** 从未登录过**
mysql	** 从未登录过**

5. 服务器的邮件日志

服务器的邮件为/var/log/messages，如果要用专业的日志分析工具来分析的话，我推荐使用 Awstats。如果公司的开发系统对邮件的要求比较低，可以配置最简单的 Sendmail 或 Postfix，通过看邮件日志里的 status 状态来判断邮件到底有没有正确发送。在配置 Nagios 服务器时，我也习惯用此日志来判断报警邮件到底有没有发送。如果对自己的 shell 水平足够有自信，也可以写脚本来收集邮件服务器的返回状态等。但专业的事情，建议还是由专业的 Awstats 工具来做，特别是邮件负载比较大时（比如，每天几百万条日志或上千万条日志），依靠人力完全不可取。

6. 输出 iptables 日志到一个指定的文件中

iptables 的 man 参考页中提到：我们可以使用 iptables 在 Linux 内核中建立、维护和检查 IP 包过滤规则表，iptables 自身的 3 个表可能已经创建，每一个表包含了很多内嵌的链，也可能包含用户自定义的链。iptables 默认把日志信息输出到/var/log/messages 文件中。不过在有些情况下（比如你的 Linux 服务器是用来作为防火墙或 NAT 路由器的），你可能需要修改日志输出的位置，通过修改或使用新的日志文件，可以帮你创建更好的统计信息，或者帮你分析网络攻击信息。下面向大家介绍如何建立一个新的日志文件/var/log/iptables.log。输出 iptables 日志信息到一个指定文件的方法如下所示：

1) 打开/etc/syslog.conf 文件。

```
# vim /etc/syslog.conf
```

2) 在文件末尾加入下面这行信息:

```
kern.warning /var/log/iptables.log
```

3) 保存和关闭文件, 使用下面的命令重新启动 syslogd。

```
/etc/init.d/syslog restart
```

7. 日志文件的专业工具

系统的一些服务, 比如 Apache、Nginx、Squid, 还有 MySQL 数据服务器, 都有自己特定的日志文件, 不过由于其格式比较复杂, 还是推荐使用专业工具 (如 Awstats、Cacti) 来分析。MySQL 的 binlog 日志可以用 mysqlbinlog 来分析, Cacti 用得比较多的情况是用来分析 Nginx 负载均衡器一段时间内的并发情况及服务器的流量异常情况。

8. 用 dmesg 查看启动消息

dmesg 提供了一个简单的方法查看系统启动信息。当 Linux 启动的时候, 内核的信息被存入内核 ring 缓存当中, dmesg 可以显示缓存中的内容。默认情况下, dmesg 打印内容到屏幕上, 当然你可以将其重定向输出到一个文件中。如果硬件损坏的话, 在 dmesg 日志里是有显示的, 可用以下命令来查看 `dmesg | grep error`, 其实看到的也就是 `/var/log/dmesg` 中的内容。

9. 关于 cron 的日志

默认情况下, Crontab 中执行的日志写在 `/var/log` 下, 我们可以先看看 `/etc/syslog.conf` 里的配置, 通过命令 `grep cron /etc/syslog.conf` 来查看, 如下所示:

```
[root@localhost log]# grep cron /etc/syslog.conf
*.info;mail.none;authpriv.none;cron.none /var/log/messages
# Log cron stuff
cron.* /var/log/cron
```

接着看 `/var/log/` 下的 cron 日志, 如下所示:

```
[root@localhost log]# ls -lsart /var/log/cron*
80 -rw----- 1 root root 72378 03-20 04:02 /var/log/cron.2
812 -rw----- 1 root root 819861 03-27 04:02 /var/log/cron.1
524 -rw----- 1 root root 525442 03-31 13:59 /var/log/cron
```

当 crond 执行任务失败时, Crontab 的日志会给用户发一封邮件。如果在服务器上发现一个任务没有正常执行, 而 crond 的邮件发送也失败, 那么就检查一下 mail 的日志, 看看是否是因磁盘空间不够而造成的。

为了方便收集 crond 的日志信息, 也可以将 cornd 错误输出和标准输出日志都指向自定义的日志文件:

```
0 6 * * * root /root/test_file.sh >> /data/log/mylog.log 2 > &1
```

10. 用 shell 或 perl 来分析日志

我们在维护线上服务器时, 并不是每台服务器的日志都需要查看, 可以偏重于我们有需求的服务器。如果不太喜欢用 Awstats 来分析 Nginx 负载均衡器的日志, 可以编写一段分析日志的脚本, 下一节我将跟大家分享用 shell 编写的分析 Nginx 日志的脚本。

1.4.3 用 shell 脚本分析 Nginx 日志

本节将介绍用 shell 脚本来分析 Nginx 负载均衡器的日志，这样可以快速得出排名靠前的网站和 IP 等，推荐大家使用线上环境下的 shell 脚本。本节中的 shell 脚本又分为两种情况，第一种情况是 Nginx 作为最前端的负载均衡器，其集群架构为 Nginx + Keepalived 时，脚本内容如下所示：

```
vim log-nginx.sh
#!/bin/bash

if [ $# -eq 0 ]; then
    echo "Error: please specify logfile."
    exit 0
else
    LOG=$1
fi

if [ ! -f $1 ]; then
    echo "Sorry, sir, I can't find this apache log file, pls try again!"
    exit 0
fi

#####
echo "Most of the ip:"
echo "-----"
awk '{ print $1 }' $LOG | sort | uniq -c | sort -nr | head -10
echo
echo
#####
echo "Most of the time:"
echo "-----"
awk '{ print $4 }' $LOG | cut -c 14-18 | sort | uniq -c | sort -nr | head -10
echo
echo
#####
echo "Most of the page:"
echo "-----"
awk '{print $11}' $LOG | sed 's/^.*\(.cn*\)/\1/g' | sort | uniq -c | sort -rn | head -10
echo
echo
#####
echo "Most of the time / Most of the ip:"
echo "-----"
awk '{ print $4 }' $LOG | cut -c 14-18 | sort -n | uniq -c | sort -nr | head -10 > timelog

for i in `awk '{ print $2 }' timelog`
do
    num=`grep $i timelog | awk '{ print $1 }'`
    echo "$i $num"
    ip=`grep $i $LOG | awk '{ print $1 }' | sort -n | uniq -c | sort -nr | head -10`
```

```

    echo "$ip"
    echo
done
rm -f timelog

```

第二种情况是以 Nginx 作为 Web 端，置于 LVS 后面，这时要剔除掉 LVS 的 IP 地址，比如 LVS 服务器的公网 IP 地址（像 203.93.236.141、203.93.236.145 等）。这样可以将第一种情况的脚本略微调整一下，如下所示：

```

#!/bin/bash
if [ $# -eq 0 ]; then
    echo "Error: please specify logfile."
    exit 0
else
    cat $1 | egrep -v '203.93.236.141|145' > LOG
fi

if [ ! -f $1 ]; then
    echo "Sorry, sir, I can't find this apache log file, pls try again!"
    exit 0
fi

#####
echo "Most of the ip:"
echo "-----"
awk '{ print $1 }' LOG | sort | uniq -c | sort -nr | head -10
echo
echo
#####
echo "Most of the time:"
echo "-----"
awk '{ print $4 }' LOG | cut -c 14-18 | sort | uniq -c | sort -nr | head -10
echo
echo
#####
echo "Most of the page:"
echo "-----"
awk '{ print $11 }' LOG | sed 's/^\.*\(.cn*\)/\1/g' | sort | uniq -c | sort -rn | head -10
echo
echo
#####
echo "Most of the time / Most of the ip:"
echo "-----"
awk '{ print $4 }' LOG | cut -c 14-18 | sort -n | uniq -c | sort -nr | head -10 > timelog

for i in `awk '{ print $2 }' timelog`
do
    num=`grep $i timelog | awk '{ print $1 }'`
    echo "$i $num"
    ip=`grep $i LOG | awk '{ print $1 }' | sort -n | uniq -c | sort -nr | head -10`

```

```

    echo "$ip"
    echo
done
rm -f timelog

```

我们可以用此脚本分析文件名为 `www_tomcat_20110331.log` 的文件。

```
[root@localhost 03]# sh counter_nginx.sh www_tomcat_20110331.log
```

大家应该跟我一样比较关注脚本运行后的第一项和第二项结果，即访问我们网站最多的 IP 和哪个时间段 IP 访问比较多，如下所示：

Most of the ip:

```

-----
5440 117.34.91.54
  9 119.97.226.226
  4 210.164.156.66
  4 173.19.0.240
  4 109.230.251.35
  2 96.247.52.15
  2 85.91.140.124
  2 74.168.71.253
  2 71.98.41.114
  2 70.61.253.194

```

Most of the time:

```

-----
12 15:31
11 09:45
10 23:55
10 21:45
10 21:37
10 20:29
10 19:54
10 19:44
10 19:32
10 19:13

```

如果对日志的要求不高，我们可以直接通过 `Awk` 和 `Sed` 来分析 Linux 日志（如果对 Perl 熟练也可以用它来操作），还可以通过 `Awstats` 来进行详细分析，后者尤其适合 Web 服务器和邮件服务器。另外，如果对日志有特殊需求的话，还可以架设专用的日志服务器来收集 Linux 服务器日志。总之一句话：一切看需求而定。

1.5 Linux 服务器的优化

服务器的优化是我们最小化安装系统时应该做的事情。其实，在做这项工作之前，我们就应该根据实际应用需求来选购 Linux 服务器，然后有所偏重地选择硬件，比如我们应该根据服务器的应用来确定是需要 RAID5，还是单块硬盘等。

1.5.1 根据服务器应用来选购服务器

无论是租用服务器还是托管都要面临一个问题，那就是选择服务器的硬件配置。前面也说了，选购硬件配置时要根据我们的服务器应用需求而定。因为你无法通过一台服务器来满足所有的需求、解决所有的问题。在项目实施或网站架构之前，我们应该从以下几方面来考虑如何选购 Linux 服务器：

- 服务器运行的应用
- 需要支持的用户访问量
- 需要的存储数据空间
- 业务的重要性
- 服务器网卡方面的考虑
- 安全方面的考虑
- 机架合理化安排
- 服务器的价格预算

下面分别来看一下。

1. 服务器运行的应用

这是首先需要考虑的问题，我们通常要根据服务器的应用类型（也就是用途）来决定服务器的性能、容量和可靠性需求。下面将按照负载均衡、缓存服务器、前端服务器、应用程序服务器、数据服务器的常见基础架构来讨论。

- 负载均衡：它对服务器的要求非常低，尤其是用来做 LVS 负载时，它会直接将所有的连接要求转给后端的 Web 应用。所以，在保证网卡性能的前提下（很多时候我用的是品牌服务器自带的百兆网卡），可以将性能相对较差的配置用作负载均衡。
- 缓存服务器：主要是 squid 或 varnish 等，需要重点考虑两方面的因素，即内存尽量多些，硬盘尽量快些，不要因为硬盘的 I/O 影响了整体性能。
- Web 前端服务器：正常情况下，大多数 Web 前端服务器（Front-end）对服务器的要求不高，例如静态 Web 服务器、动态服务器、图片服务器等。事实上现在很流行在一台性能卓越的服务器上同时运行 Web 前端 + 应用服务器，比如 Nginx + PHP - FPM、Nginx + Tomcat 或 Nginx + Resin。
- 应用程序服务器：由于它承担了计算和功能实现的重任，我们需要为基于 Web 架构的应用程序服务器（Application Server）选择足够快的服务器。另外应用程序服务器可能需要用到大量的内存，尤其是基于 Windows 基础架构的 Ruby、Python、Java 服务器，这一类服务器至少需要使用单路至强的配置。至于可靠性的问题，如果你的架构中只有一台应用服务器，那这台服务器需要足够可靠，RAID 是绝对不能忽视的选项。但如果有两台或更多的应用服务器，并设计了负载均衡机制、具有冗余功能，那就不必过于担心了。
- 特殊的应用：除了用于 Web 架构中的应用程序之外，如果你的服务器还要处理流媒体视频编码、服务器虚拟化、媒体服务器（Asterisk 之类），或者作为游戏服务器（逻辑、地图、聊天）运行，那同样会对 CPU 和内存的需求比较高，至少也要考虑单路至强的服务器。其中，服务器虚拟化对存储可靠性的要求非常高，因为一个篮子里有十几个鸡蛋，篮子一定要足够牢靠才行。

- 公共服务：这里指的是邮件服务器、文件服务器、DNS 服务器、域控服务器等。通常我们会部署两台 DNS 服务器互相备份，域控主服务器也会拥有一台备份服务器（专用的或非专用的），所以对于可靠性无须过于苛刻。至于邮件服务器，至少需要具备足够的硬件可靠性和容量大小，这主要是对邮件数据负责，因为很多用户没有保存和归档邮件数据的习惯，待其重装系统后，就会习惯性地到服务器上重新下载相应的数据。至于性能问题，则应评估用户数量后再决定。另外，考虑到它的重要性，建议还要尽量选择稳定的服务器系统，比如 Linux 或 BSD 系列。
- 数据库：这是我们最后讨论的应用，对服务器的要求也是最高、最重要的。无论你使用的是 MySQL、SQL Server 还是 Oracle，一般情况下，它都需要有足够快的 CPU、足够大的内存、足够稳定可靠的硬件。可直接采用 Dell PowerEdge R710 或 HP 580G5，CPU 和内存也要尽可能最大化。如果预算充分，建议用固态硬盘作为 RAID10，因为数据库服务器对硬盘的 I/O 要求是最高的。

2. 服务器需要支持的用户访问量

服务器就是为了给用户某种服务的，所以使用这些服务的用户同样是我们必须考虑的因素。我们可以从下面几个具体的问题进行评估：

有多少注册用户？正常情况下有多少用户会同时在线访问？每天同时在线访问的最高峰值大概是多少？

一般在项目实施之前，客户方面会针对这些问题给出一个大致的结果。但我们要尽量设计得比这更充分和具体。同时，我们还要对未来的用户增长做一个尽可能准确的预测和规划，因为你的服务器可能会支持越来越多的用户，所以在进行网站或系统架构时要让机器能灵活地扩展。

3. 需要的存储数据空间

关于这个问题需要从两个方面来考虑，一方面是有哪些类别的数据，包括：操作系统本身占用的空间，安装应用程序所需要的空间，应用程序所产生的数据、数据库、日志文件、邮件数据等，如果网站是 Web 2.0 的，还要计算每个用户的存储空间；另一方面是从时间轴上来考虑，这些数据每天都在增长，你至少要为未来 1 年（我们建议 2~3 年）的数据增长做个准确的测算，这就需要软件开发人员和业务人员一起来提供足够的信息了。最后可将计算出来的结果乘上 1.5 左右的系数，方便维护的时候做各种数据的备份和文件转移操作。

4. 业务的重要性

关于这个问题就需要根据自身的业务领域来考虑相关要求了。下面举几个简单的例子，帮助你了解这些服务器对可靠性、数据完整性等方面的要求。

- 如果你的服务器是用来运行一个 WordPress 博客、与朋友们分享观点的，那么，一台酷睿服务器、1GB 的内存外加一块 160GB 的硬盘就足够了。就算服务器出现了一点硬件故障，导致几个小时甚至一两天不能提供访问，生活会照常继续，天也不会塌下来。
- 如果你的服务器是用作测试平台的，那么就不会像生产环境那样对可靠性有极高的要求，你所需要的可能只是做好例行的数据备份，若服务器宕机，只要能在当天把问题解决就行了。
- 如果是一个电子商务公司的服务器，运行着电子商务网站平台，那么请一定要十分重视服务

器。当硬件发生故障而导致宕机时，你需要对以下“危言耸听”的后果做好心理准备：投诉电话被打爆、顾客大量流失、顾客要求退款、市场推广费用打水漂、员工无事可干、公司运营陷入瘫痪状态、数据丢失等。事实上，电子商务网站一般是需要 365 × 24 小时不间断监控的，而且要有专人轮流值守，并且要有足够的备份设备，每天还要有专人检查。个人感觉，在网站运维方面，电子商务的技术含量也是最高的。

- 如果是大型广告类或门户类网站，那么建议选择 CDN 系统。由于它有提高网站响应速度、负载均衡、有效抵御 DDoS 攻击等特点，相对而言，每节点都会有大量的冗余，所以，除了成本之外，CDN 机器的硬盘问题不大。

这里其实只是简单地讨论了业务对服务器硬件可靠性的要求。换言之，如果你觉得业务不能承担硬盘损坏带来的停机或数据丢失风险，那么一定要选择一个合适的 RAID 卡。对于冗余电源问题，道理是一样的（要全面解决这个问题，不能只考虑单个服务器的硬件，还需要结合系统架构的规划设计）。

在回答了以上问题后，接下来就可以决定下面这些具体选项了：

（1）选择什么 CPU

回忆一下上面关于“服务器运行的应用”和“需要支持的用户访问量”两个方面的考虑，这将帮助我们选择合适的 CPU。毫无疑问，CPU 的主频越高，其性能也就越高；两个 CPU 要比一个 CPU 来得更爽，至强肯定比酷睿更猛。但究竟怎样的 CPU 才是合适的呢？下面为你提供一些常见情况下的建议：

- 如果你的业务刚刚起步，预算不是很充足，建议你选择一款经典的酷睿服务器，这可以帮你节约大量成本。而且，以后可以根据业务发展的情况，随时升级到更高配置的服务器。
- 如果你需要在一台服务器上同时运行多种应用服务，例如 .Net + Exchange + SQL Server，那么一个单核至强（例如 X3330）或新一代的酷睿 I3/I5（双核四线程）将是最佳的选择。虽然从技术的角度来说，这不是一个好主意，但至少能够帮你节约一大笔成本。
- 如果你的服务器要运行 SQL Server、MySQL 或 Oracle，而且目前几百个用户同时在线，未来还会不断增长，那么你至少应该选择安装一个双四核服务器。
- 如果需要的是 Web 应用服务器，双四核基本就可以满足我们的要求了。

（2）需要多大的内存

同样，“服务器运行的应用”和“需要支持的用户访问量”两方面的考虑也将帮助我们选择合适的内存容量。相比于 CPU，我认为内存（RAM）才是影响性能的最关键因素。因为在相当多正在运行的服务器中，CPU 的利用率一般都在 10% ~ 30% 之间，甚至更低。但我们发现由于内存容量不够而导致服务器运行缓慢的案例比比皆是，如果服务器不能分配足够的内存给应用程序，应用程序就需要通过硬盘接口缓慢地交换读写数据了，这将导致网站慢得令人无法接受。内存的大小主要取决于服务器的用户数量，当然也和应用软件对内存的最低需求及内存管理机制有关，所以，最好由程序员或软件开发商给出最佳的内存配置建议。下面同样给出了一些常见应用环境下的内存配置建议：

- 无论是 Windows 下的 IIS 还是 Linux 下的 Apache，一般情况下 Web 前端服务器不需要配置特别高的内存，尤其是在集群架构中，4GB 的内存就已经足够了。如果有几千个并发用户，而且他们同时运行动态脚本程序，我们才会考虑使用 8GB 或更高的内存。

- 对于运行 Tomcat、Resin、WebLogic、Websphere 或 .NET 的应用服务器，4GB 内存应该是基准配置，更准确的数字需要根据用户数量和技术架构来确定。
- 数据库服务器的内存由数据库实例的数量、表大小、索引、用户数来决定，一般建议配置 4GB 以上的内存，我们在许多项目方案中使用了 24GB ~ 48GB 的内存。
- 诸如 Postfix、Notes、Exchange 这样的邮件服务器对内存的要求并不高，1GB ~ 2GB 就可以满足了。
- 还有一些特殊的服务器，我们需要为之配置尽可能高的内存容量，包括 Squid、Varnish、Memcached 的缓存服务器。
- 对于一台文件服务器，1GB 内存可能就足够了。

事实上，上面的数字已经足够“慷慨”，由于内存技术在不断进化，价格也在不断降低，我们才得以近乎奢侈地讨论 4GB、8GB、16GB 这些曾经不可想象的内存容量。然而，除了花钱购买内存来满足应用程序的“贪婪”之外，系统优化和数据库优化仍然是我们需要重视的问题。

(3) 需要怎样的硬盘存储系统

硬盘存储系统的选择和配置是整个服务器系统里最复杂的一部分，我们需要考虑硬盘的数量、容量、接口类型、转速、缓存大小，以及是否需要 RAID 卡、RAID 卡的型号和 RAID 级别等问题。甚至在一些高可靠性、高性能的应用环境中，我们还需要考虑使用怎样的外部存储系统（SAN、NAS 或 DAS）。下面将服务器的硬盘 RAID 卡的特点归纳一下：

- 如果是用作缓存服务器，比如 squid、varnish 还有 memcached，可以考虑用 RAID0。
- 如果是运行 Nginx + PHP5 或 tomcat、resin 等应用，可以考虑用 RAID1。
- 如是是内网开发服务器或存放重要代码的服务器，可以考虑用 RAID5。
- 如果是运行 MySQL 或 Oracle 等数据库应用，可以考虑用固态硬盘做 RAID5 或 RAID10。

5. 网卡性能与数据方面的考虑

如果你的基础架构是多服务器环境，而且服务器之间有大量的数据交换，那么建议你为每台服务器配置两个或更多的网卡，一个用来对外提供服务，另一个用来做内部数据交换。由于现在项目外端都置于防火墙内，所以许多时候单网卡就足够了；而像 LVS + Keepalived 这种只用公网地址的 Linux 集群架构，有时可能仅仅需要一块网卡。目前，HP 或 Dell 这种品牌服务器自带的网卡已经足够使用了。

另外，数据的备份也是很重要的。在实际工作中我们也发现，rsync 和 scp 这些 Linux 下的备份工具同样非常占带宽，所以，如果用 scp，建议尽量用它的限速参数；而 rsync 则尽量选择在非业务时间段执行。

6. 服务器安全方面的考虑

由于目前国内的 DDoS 攻击还是比较普遍，建议给每个项目方案和自己的电子商务网站配备硬件防火墙，比如 Juniper、Cisco 或神盾等。当然了，这个问题也是网站后期运营维护需要考虑的，这里只是想让大家有个概念性的认识。有时为了数据的安全，我会让所有的机器都用 RAID5。另外就是定期巡视机房，检查服务器的硬盘灯指向，一有异常就迅速处理。

7. 根据机架数合理安排服务器的数量

这个问题应该在项目实施前就准备好，选择服务器时应该明确 1U、2U 和 4U 到底有多少台，应该如何安排。在小项目中这个问题可能无关紧要，但在大型项目的实施过程中，这个问题就很突

出了，我们应该根据现有或额定的机架数目确定到底应该选择多少个服务器。

8. 成本考虑：服务器的价格问题

这个问题无论是在替公司采购时，还是在项目实施过程中，都是重要的问题。我们的方案经常被退回，理由就是超出预算。尤其是一些小项目，预算更吃紧。我做项目时经常面临的一种需求是客户做的是证券类资讯网站，只要求周一至周五的上午9点至下午3点网站不出问题即可，并不想做复杂的负载均衡高可用。所以这时候，我会做成单 Nginx 或 Haproxy，后面接两台 Web 应用。这种情况还好说，如果是做中大型电子商务网站，在服务器成本上的控制就尤其重要。事实上，我们经常遇到的问题是，客户给出的成本预算有限，而我们的应用又需要更多的服务器。这时候，我们不得不选择 Centos 或 FreeBSD 下的免费虚拟化软件，这将在后面的章节中重点讲述。

以上8个方面即是我们在采购服务器时应该注意的因素，在选择服务器的组件时要有所偏重，然后根据系统或网站架构来决定服务器的数量，尽量做到服务器资源利用的最大化。

1.5.2 Centos5.5 最小化安装后的优化

购买了服务器（现在主流配置都是双四核），下一步就要安装系统了。这里推荐用 64 位的 Centos5.5，安装系统时我们要选择最小化安装（不要图形）。大家在用服务器时要记得一个原则，系统的安装包越少越好，这样机器才会更稳定。前面已经介绍过线上服务器的分区流程，如果遇到对磁盘 I/O 调用频繁的服务（例如 MySQL），我们可以单独拿一个分区（如 /data）出来，不要跟 / 装在一起，避免 / 分区被频繁调用，出现 I/O 瓶颈。至于单服务器的性能调优，本着稳定安全的原则，尽量不要改动系统原有的配置（Centos 自身的文件和内存机制就很优秀），尤其是线上环境，稳定性要放在第一位来考虑。

1. 关闭不需要的服务

众所周知，服务越少，系统占用的资源也会越少，所以应关闭不需要的服务。这样做的好处是减少内存和 CPU 时间的占用。命令如下所示：

```
# ntsysv
```

下面列出需要启动的服务，未列出的服务一律关闭。

☐ crond

Linux 下的时间计划任务服务。

☐ irqbalance

启用 irqbalance 服务，既可以提升性能，又可以降低能耗。irqbalance 用于优化中断分配，它会自动收集系统数据以分析使用模式，并依据系统负载状况将工作置于 Performance mode 或 Power-save mode 状态。处于 Performance mode 时，irqbalance 会将中断尽可能均匀地分发给各个 CPU core，以充分利用 CPU 的多核，提升性能。处于 Power-save mode 时，irqbalance 会将中断集中分配给第一个 CPU，以保证其他空闲 CPU 的睡眠时间，降低能耗。现在的主流服务器都是双四核，所以这项我建议保留。

☐ network

☐ sshd

☐ syslog

这是 Linux 的日志系统，必须要启动，否则机器出现问题时会找不到原因。

再说一下两个比较特殊的服务，它们是 iptables 和 SELinux。因为网站和系统之前均有硬件防火墙，如果没有特殊需求的话，均可选择关闭。要关闭它们可以在命令 setup 下操作，也可以使用命令行操作。

关闭 iptables 的代码如下：

```
service iptables stop && chkconfig iptables off
```

关闭 SELinux 的方法如下：

```
vim /etc/selinux/config
```

然后将文件中的 selinux = "" 改为 disabled，并重启。如果不想重启系统，使用命令 setenforce 0，此命令可以暂时关闭 SELinux，重启后失效。

说明 setenforce 1 将 SELinux 设置成为 enforcing 模式；setenforce 0 将 SELinux 设置成为 permissive 模式。

另外，在 lilo 或 grub 的启动参数中增加：selinux = 0，这样也可以关闭 SELinux。

2. 关闭不需要的 tty

先编辑 /etc/inittab，找到如下一段命令：

```
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6
```

这段命令会使 init 为你打开了 6 个控制台，分别可以用 [ALT + F1] 到 [ALT + F6] 进行访问。此 6 个控制台默认都驻留在内存中，用 ps -aux 这个命令就可以看到，这是 6 个进程，如下所示：

```
[root@localhost ~]# ps -aux | grep tty
Warning: bad syntax, perhaps a bogus '- '? See /usr/share/doc/procps-3.2.7/FAQ
root      3219  0.0  0.0  3792  488 tty1      Ss+  Mar16   0:00 /sbin/mingetty tty1
root      3220  0.0  0.0  3792  484 tty2      Ss+  Mar16   0:00 /sbin/mingetty tty2
root      3221  0.0  0.0  3792  488 tty3      Ss+  Mar16   0:00 /sbin/mingetty tty3
root      3222  0.0  0.0  3792  488 tty4      Ss+  Mar16   0:00 /sbin/mingetty tty4
root      3224  0.0  0.0  3792  488 tty5      Ss+  Mar16   0:00 /sbin/mingetty tty5
root      3226  0.0  0.0  3792  488 tty6      Ss+  Mar16   0:00 /sbin/mingetty tty6
root      3325  0.0  0.1  90548  6264 tty7      Ss+  Mar16   0:01 /usr/bin/Xorg :0 -br -audit 0
- auth /var/gdm/:0.Xauth -nolisten tcp vt7
root      6767  0.0  0.0  68284  1564 tty8      Ss+  Mar17   0:00 /bin/bash
root      31179  0.0  0.0  63372   756 pts/2    S+   17:03   0:00 grep tty
```

事实上没有必要使用这么多。应如何关闭不需要的进程呢？通常我们保留前两个控制台就可以了，把后面 4 个用 # 注释掉，并且无需重启，只需要执行 init q 这个命令即可，如下所示：

```
init q
```

3. 对 TCP/IP 网络参数进行调整

调整 TCP/IP 网络参数，可以增强抗 SYN Flood 的能力，命令如下所示：

```
# echo 'net.ipv4.tcp_syncookies = 1' >> /etc/sysctl.conf
# sysctl -p
```

操作前建议用 `sysctl -a > sysctl -file` 保存下当前配置。

4. 修改 shell 命令的 history 记录个数

修改 history 记录的命令如下所示：

```
# vi /etc/profile
```

找到 `histsize = 1000`，将其改为 `histsize = 100`（这条可根据实际情况而定）。不重启系统就可让其生效，如下所示：

```
source /etc/profile
```

5. 定时校正服务器的时间

我们可以定时校正服务器的时间，命令如下所示：

```
# yum install ntp
# crontab -e
```

加入如下一行：

```
*/5 * * * * ntpdate ntp.api.bz
```

#ntp.api.bz 是一组 NTP 服务器集群，目前有 6 台服务器。这项服务是 api.bz 继 `http://sms.api.bz` 移动飞信免费短信发送接口之后的第二项免费 API 服务。

6. 停止打印服务

如果不准备提供打印服务，可停止默认设置为自动启动的打印服务，命令如下所示：

```
[root@sample ~]# /etc/rc.d/init.d/cups stop ← 停止打印服务 Stopping cups: [OK] ← 停止服务成功,出现“OK”
```

```
[root@sample ~]# chkconfig cups off ← 禁止打印服务自动启动
```

```
[root@sample ~]# chkconfig --list cups ← 确认打印服务自启动设置状态
```

```
cups0:off 1:off 2:off 3:off 4:off 5:off 6:off ← 0~6 都为 off 的状态就 OK (当前打印服务自启动被禁止)
```

7. 停止 ipv6

在 Centos5.5 默认的状态下，ipv6 是被启用的。因为我们不使用 ipv6，所以，可以停止 ipv6，以最大限度地保证安全和快速。首先确认一下 ipv6 是不是处于被启动的状态。

```
[root@sample ~]# ifconfig -a ← 列出全部网络接口信息
eth0 Link encap:Ethernet HWaddr 00:0C:29:B6:16:A3
inet addr:192.168.0.13 Bcast:192.168.0.255 Mask:255.255.255.0
inet6 addr: fe80::20c:29ff:feb6:16a3/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:84 errors:0 dropped:0 overruns:0 frame.:0
TX packets:93 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:10288 (10.0 KiB) TX bytes:9337 (9.1 KiB)
```

```

Interrupt:185 Base address:0x1400

lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:16436 Metric:1
RX packets:12 errors:0 dropped:0 overruns:0 frame.:0
TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:952 (952.0 b) TX bytes:952 (952.0 b)
sit0 Link encap:IPv6 - in - IPv4 ← 确认 ipv6 是被启动的状态
NOARP MTU:1480 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame.:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:0 (0.0 b) TX bytes:0 (0.0 b)

```

然后修改相应配置文件，停止 ipv6，如下所示：

```

[root@sample ~]#vi /etc/modprobe.conf ← 修改相应配置文件,添加如下内容
alias net-pf-10 off
alias ipv6 off
echo "IPV6INIT=no" >> /etc/sysconfig/network-scripts/ifcfg-eth0
[root@sample ~]#shutdown -r now ← 重新启动系统,使设置生效

```

最后确认 ipv6 的功能已经被关闭，如下所示：

```

[root@sample ~]#ifconfig -a ← 列出全部网络接口信息
eth0 Link encap:Ethernet HWaddr 00:0C:29:B6:16:A3
inet addr:192.168.0.13 Bcast:192.168.0.255 Mask:255.255.255.0
inet6 addr: fe80::20c:29ff:feb6:16a3/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:84 errors:0 dropped:0 overruns:0 frame.:0
TX packets:93 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:10288 (10.0 KiB) TX bytes:9337 (9.1 KiB)
Interrupt:185 Base address:0x1400lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:16436 Metric:1
RX packets:12 errors:0 dropped:0 overruns:0 frame.:0
TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:952 (952.0 b) TX bytes:952 (952.0 b)

```

确认 ipv6 的相关信息没有被列出，说明 ipv6 功能已经关闭。

8. 调整 Linux 的最大文件打开数

要调整一下 Linux 的最大文件打开数，否则 squid 在高负载时执行性能将会很低。另外，在 Linux 下面部署应用时，有时候会遇上 Socket/File: Can't open so many files 这样的问题，这个值也会影响服务器的最大并发数。其实 Linux 是有文件句柄限制的，但默认不是很高，一般是 1024，生

产服务器很容易就会达到这个值，所以需要改动此值。刚开始我采用 `vim /etc/security/limit.conf` 命令，在最后一行添加如下代码：

```
* soft nofile 60000
* hard nofile 65535
```

但重启后一切都还原了。

正解做法应该为在 Centos5.5 的 `/etc/rc.local` 文件里添加如下命令行：

```
ulimit -SHn 65535
```

当然了，我们也可以在 Nginx 的一些监控脚本里实时添加此命令行，达到重启也能生效的目的。

另外，`ulimit -n` 命令并不能真正看到文件的最大文件打开数，大家可用如下脚本查看：

```
#!/bin/bash
for pid in `ps aux |grep nginx |grep -v grep |awk '{print $2}'`
do
cat /proc/${pid}/limits |grep 'Max open files'
done
```

9. 启动网卡

大家配置 Centos5.5 的网卡时，容易忽略的一项就是 Linux 启动时未启动网卡，其后果很明显，那就是你的 Linux 机器永远也没有 IP 地址，下面是一台线上服务器的配置：

```
[root@localhost ~]# vim /etc/sysconfig/network-scripts/ifcfg-eth0
# Intel Corporation 82541GI Gigabit Ethernet Controller
DEVICE=eth0
BOOTPROTO=none
HWADDR=00:14:22:1B:71:20
IPV6INIT=yes
IPV6_AUTOCONF=yes
ONBOOT=yes →此项一定要记得为 yes,它会在系统引导就启动你的网卡设备
NETMASK=255.255.255.192
IPADDR=203.93.236.146
GATEWAY=203.93.236.129
TYPE=Ethernet
PEERDNS=yes →允许从 DHCP 处获得的 DNS 覆盖本地的 DNS
USERCTL=no ~ →不允许普通用户修改网卡
```

10. 关闭 Centos5.5 的写磁盘 I/O 功能

一个 Linux 文件默认有 3 个时间。

- ☐ atime：对此文件的访问时间。
- ☐ ctime：此文件 inode 发生变化的时间。
- ☐ mtime：此文件的修改时间。

如果有多个小文件（比如 Web 服务器的页面上有多个小图片），通常就没有必要记录文件的访问时间了，这样可以减少写磁盘的 I/O。这要如何配置呢？

首先，修改文件系统的配置文件：`vim /etc/fstab`。然后，在包含大量小文件的分区中使用

noatime和 nodiratime 这两个命令。例如：

```
/dev/sda5 /data/pics ext3 noatime,nodiratime 0 0
```

这样文件被访问时就不会再产生写磁盘的 I/O 了，此方法尤其适合读写频繁的数据库系统。

1.5.3 优化 Linux 下的内核 TCP 参数以提高系统性能

内核的优化跟服务器的优化一样，应本着稳定安全的原则。下面以 64 位的 Centos5.5 下的 Squid 服务器为例来说明，待客户端与服务器端建立 TCP/IP 连接后就会关闭 SOCKET，服务器端连接的端口状态也就变为 TIME_WAIT 了。那是不是所有执行主动关闭的 SOCKET 都会进入 TIME_WAIT 状态呢？有没有什么情况使主动关闭的 SOCKET 直接进入 CLOSED 状态呢？答案是主动关闭的一方在发送最后一个 ACK 后就会进入 TIME_WAIT 状态，并停留 2MSL (Max Segment LifeTime) 时间，这个是 TCP/IP 必不可少的，也就是“解决”不了的。

TCP/IP 的设计者如此设计，主要原因有两个：

- 防止上一次连接中的包迷路后重新出现，影响新的连接（经过 2MSL 时间后，上一次连接中所有重复的包都会消失）。
- 为了可靠地关闭 TCP 连接。主动关闭方发送的最后一个 ACK (FIN) 有可能会丢失，如果丢失，被动方会重新发 FIN，这时如果主动方处于 CLOSED 状态，就会响应 RST 而不是 ACK。所以主动方要处于 TIME_WAIT 状态，而不能是 CLOSED 状态。另外，TIME_WAIT 并不会占用很大的资源，除非受到攻击。

在 Squid 服务器中可输入查看当前连接统计数的命令，如下所示：

```
netstat -n|awk '/^tcp/ {++S[$NF]} END {for(a in S) print a, S[a]}'
LAST_ACK 14
SYN_RECV 348
ESTABLISHED 70
FIN_WAIT1 229
FIN_WAIT2 30
CLOSING 33
TIME_WAIT 18122
```

- CLOSED：无连接是活动的或正在进行中的。
- LISTEN：服务器在等待进入呼叫。
- SYN_RECV：一个连接请求已经到达，等待确认。
- SYN_SENT：应用已经开始，打开一个连接。
- ESTABLISHED：正常数据传输状态。
- FIN_WAIT1：应用说它已经完成。
- FIN_WAIT2：另一边已同意释放。
- CLOSING：两边同时尝试关闭。
- TIME_WAIT：另一边已初始化一个释放。
- LAST_ACK：等待所有分组死掉。

也就是说，这条命令可以把当前系统的网络连接状态分类汇总。

在 Linux 下高并发的 Squid 服务器中，TCP TIME_WAIT 套接字数量经常可达两三万，服务器很

容易就会被拖死。不过，我们可以通过修改 Linux 内核参数来减少 Squid 服务器的 TIME_WAIT 套接字数量，命令如下所示：

```
vim /etc/sysctl.conf
```

然后，增加以下参数：

```
net.ipv4.tcp_fin_timeout = 30
net.ipv4.tcp_keepalive_time = 1200
net.ipv4.tcp_syncookies = 1
net.ipv4.tcp_tw_reuse = 1
net.ipv4.tcp_tw_recycle = 1
net.ipv4.ip_local_port_range = 1024 65000
net.ipv4.tcp_max_syn_backlog = 8192
net.ipv4.tcp_max_tw_buckets = 5000
```

其中：

- net.ipv4.tcp_syncookies = 1 表示开启 SYN Cookies。当出现 SYN 等待队列溢出时，启用 cookie 来处理，可防范少量的 SYN 攻击。默认为 0，表示关闭。
- net.ipv4.tcp_tw_reuse = 1 表示开启重用。允许将 TIME-WAIT 套接字重新用于新的 TCP 连接。默认为 0，表示关闭。
- net.ipv4.tcp_tw_recycle = 1 表示开启 TCP 连接中 TIME-WAIT 套接字的快速回收。默认为 0，表示关闭。
- net.ipv4.tcp_fin_timeout = 30 表示如果套接字由本端要求关闭，这个参数决定了它保持在 FIN-WAIT-2 状态的时间。
- net.ipv4.tcp_keepalive_time = 1200 表示当 keepalive 启用时，TCP 发送 keepalive 消息的频度。默认是 2 小时，这里改为 20 分钟。
- net.ipv4.ip_local_port_range = 1024 65000 表示向外连接的端口范围。默认值很小：32 768 ~ 61 000，改为 1024 ~ 65 000。
- net.ipv4.tcp_max_syn_backlog = 8192 表示 SYN 队列的长度，默认为 1024，加大队列长度为 8192，可以容纳更多等待连接的网络连接数。
- net.ipv4.tcp_max_tw_buckets = 5000 表示系统同时保持 TIME_WAIT 套接字的最大数量，如果超过这个数字，TIME_WAIT 套接字将立刻被清除并打印警告信息。默认为 180 000，改为 5000。对于 Apache、Nginx 等服务器，前面介绍的几个参数已经可以很好地减少 TIME_WAIT 套接字数量，但是对于 Squid 来说，效果却不大。有了此参数就可以控制 TIME_WAIT 套接字的最大数量，避免 Squid 服务器被大量的 TIME_WAIT 套接字拖死。

执行以下命令使内核配置立即生效：

```
/sbin/sysctl -p
```

如果是用于 Apache 或 Nginx 等的 Web 服务器，或 Nginx 的反向代理，则只需要更改以下几项即可：

```
net.ipv4.tcp_syncookies = 1
net.ipv4.tcp_tw_reuse = 1
net.ipv4.tcp_tw_recycle = 1
net.ipv4.ip_local_port_range = 1024 65000
```

执行以下命令使内核配置立即生效：

```
/sbin/sysctl -p
```

如果是邮件服务器，则建议内核方案如下：

```
net.ipv4.tcp_fin_timeout = 30
net.ipv4.tcp_keepalive_time = 300
net.ipv4.tcp_tw_reuse = 1
net.ipv4.tcp_tw_recycle = 1
net.ipv4.ip_local_port_range = 5000 65000
kernel.shmmax = 134217728
```

执行以下命令使内核配置立即生效：

```
/sbin/sysctl -p
```

当然这些都只是最基本的更改，大家还可以根据自己的需求来更改内核的设置，同样也要本着稳定的原则，如果服务器不稳定的话，一切工作和努力都会白费。如果以上优化仍无法满足你的要求，有可能你需要定制你的服务器内核或升级服务器硬件。至于服务的配置优化，超出了本章的内容，大家可根据自己的需求有针对性地进行更改。

1.5.4 生产服务器应尽量选择编译安装软件包

建议在安装线上的生产服务器软件包时都用源码安装，这是因为源码安装可以自行调整编译参数，最大化地定制安装结果。这里以 MySQL5.0 线上环境的编译安装来说明之，其编译参数如下所示：

```
./configure --prefix=/usr/local/mysql --without-debug --without-bench --enable-thread-safe-client --enable-assembler --enable-profiling --with-mysqld-ldflags=-all-static --with-client-ldflags=-all-static --with-charset=latin1 --with-extra-charset=utf8,gbk --with-innodb --with-csv-storage-engine --with-federated-storage-engine --with-mysqld-user=mysql --without-embedded-server --with-server-suffix=-community --with-unix-socket-path=/usr/local/mysql/sock/mysql.sock && make && make install
```

在安装 MySQL 时，源码安装与 rpm 安装相较，其特点如下：

- 我们可以针对自己的硬件平台选用合适的编译器来优化编译后的二进制代码。
- 根据不同的软件平台环境调整相关的编译参数（源码安装不仅适用于 rhel/centos，其他系统像 FreeBSD、Solaris、Ubuntu 等一样适用）。
- 可针对特定的应用场景选择需要什么组件或不需要什么组件。
- 同一台主机上可以安装多个 MySQL（rpm 安装则仅能安装一个 MySQL）。
- 根据需要存储的数据内容选择只安装需要的字符集。

理论上源码静态编译方式安装效率会高一点，但到底比 rpm 方式高多少得看具体情况，一般在 5% 左右。

某次在线上环境工作，我用 yum 卸载一个软件包时遇到了极其危险的情况，这里也跟大家分享一下：当时，yum remove 自动卸载了许多这个软件包自身依赖的系统包，导致系统崩溃了，SSH 不能登录，并且 SCP 及 RSYNC 也都不能用了，幸亏 FTP 命令还能用，所以数据很快被 FTP 传到其他服务器上了，但系统已经彻底崩溃，只有重装了。如果是源码安装的软件包就没有这种危险，需要

卸载时只需要删除软件包安装的目录即可。

综上所述，源码安装的好处如下：

- 最大的好处就是可以自行调整编译参数，最大化地定制安装结果。
- 源码安装可以选择最新的软件包，而 Linux 系统（包括 FreeBSD）自带的软件包一般都是最稳定的版本，但不能保证是最新的。
- 相对而言，源码安装的性能是最优异的。
- 源码包安装的软件卸载时极为方便和简单，更重要的是，它比较安全，尤其是对线上的生产环境而言。
- 迁徙也比较方便，如果不涉及系统库文件，复制到另一台机器上也可以使用。

这里也简单介绍一下在 Linux/Unix 下安装软件的源码三部曲，在后面我们会经常看到它们的身影。

```
./configure
make
make install
```

`./configure` 是用来检查环境变量及配置编译选项的，`make` 是用来将源代码编译成二进制文件的，而 `make install` 则会将 `make` 编译出来的文件安装到指定位置（或默认位置）。

在本节中，我们从服务器的硬件选择、安装及内核等方面对单机服务器的性能进行了优化，不过对于网站和系统来说，单机优化对整体性能提升的作用毕竟有限，整体性能提升主要靠服务器的高可用和高扩展性来实现，这部分内容在后面的章节中再来说明。

1.6 用开源工具 Nagios 监控 Linux 服务器

1.6.1 Centos5.5 下的监控工具简介

在开源系统 Centos5.5 下有许多监控工具，比如实时监控系统状态的 Nagios，还有监控网络流量的 Cacti 和 MRTG，以及我个人比较喜欢的 NTOP 和 Iptraf。另外，在 Centos5.5 下也有许多强大的命令行可用于监控系统状态，大家可以 Google 或在 51cto.com 上了解一下其具体用法。在后面的章节中，我会向大家介绍网络运维和项目实施中使用 Nagios 的一些经验和心得，希望对大家有帮助。

Centos5.5 下强大的命令行工具列表如表 1-1 所示。

其中大部分命令已在 1.2 节讲过，这里就不再重复了。

表 1-1 Linux/Unix 下的性能监控工具

工具	Most useful tool function
uptime	Average system load
dmesg	Hardware/system information
top	Processor activity
iostat	Average CPU load, disk activity
vmstat	System activity
sar	Collect and report system activity
KDE System Guard	Real time systems reporting and graphing
free	Memory usage
traffic-vis	Network monitoring (SUSE LINUX Enterprise Server only)
pmap	Process memory usage
strace	Programs
ulimit	System limits
mpstat	Multiprocessor usage

1.6.2 Nagios 应该监控的服务器基础选项

经过实践,我们认为 Nagios 应该监控的服务器参数有如下几个方面。

1) 主机存活的状态: Nagios 是用 ping 来实现的,我们在实际使用中发现,如果服务器开启了禁 ping 也没什么影响。因为如果服务器宕机的话, Nagios 会以一小时为限拼命地发邮件和短信来报告。

2) 系统 load 值: 系统 load 值的最关键含义是 CPU 运行中等待的数量,它从侧面反映了 CPU 的繁忙程度,只不过 load 值并不直接等于等待队列中的进程数量。

3) CPU 使用率: CPU 的使用率和系统 load 值一样,从另一个角度反映出 CPU 的总体繁忙情况,只不过它所提供的信息更为详细,如当前空闲的 CPU 比率、系统占用的 CPU 比率、用户进程占用的 CPU 比率、处于 I/O 等待的 CPU 比率等。

4) 磁盘 I/O 量: 磁盘 I/O 直接反映了系统硬盘繁忙情况,特别是对于数据库这种以 I/O 操作为主的系统来说, I/O 的负载将直接影响到系统的整体响应速度(尤其是作为数据库服务器而言,监控的意义更为重大)。

5) swap 进出量: swap 的使用主要体现了系统在物理内存不够的情况下使用虚拟内存的情况。有人在观察内存情况时容易走入误区,在 free -m 中应该关注 swap,而非 free。

6) 网络流量: 对数据库系统来说,网络流量也是一个不容忽视的监控点,毕竟数据库系统的数据进出量比普通服务器要大很多。当然,如果是非数据库服务器,网络流量成为瓶颈的可能性还是比较少的。

7) 僵死进程的监控: 这项指标的监控意义我就不多说了,而且这在 Nagios 中已经作为系统默认内容存在了。

8) 在 LVS + Keepalived 或 Nginx + Keepalived 中,由 Keepalived 产生的 VIP 应该重点监控,毕竟许多系统和网站都是将此内网地址由防火墙映射成为公网地址的,相当于整个网站或系统的入口,其重要性就不言而喻了。另外,如果网站架构里有 DRBD + Heartbeat 的话,也强烈建议监控 DRBD 及 Heartbeat 的状态,因为这个牵涉到的是文件服务器,牵涉到网站最重要的图片和代码。

9) 数据库 MySQL 的主从复制状态,尤其是当数据库采用 Dual-Master-Slaves 读写分离架构时。大家应该知道,MySQL 的复制是异步的,所以 MySQL Slave 这个监控就很重要了。在工作中我们也发现,如果 MySQL 的复制出了问题,对网站的业务会很有影响,尤其是门户型社交网站。

1.6.3 Nagios 监控 Windows 2003 时应注意的事项

我在工作中发现,部分系统管理员对 Windows 2003 服务器不够重视,但是就我目前接触的金融系统和游戏运维来看,大家都是运行 Windows 2003 (以下均指 64 位的 R2 版) + SQL Server 2008,而且效果也比想象的要好。我公司的邮件系统和业务平台也是以 Windows 2003 (Windows 2003 + 域控 + Exchange 2007) 为平台的,如果服务器崩溃,会给公司带来直接经济损失。以下注意事项为工作经历和项目实施中的总结:

1) Windows 2003 客户端的 12489 端口必须开放(因为它通过此端口与 Nagios 服务器进行通信的)。可以在 Windows 2003 中用命令 netstat -an 来观察此端口到底开放没有。注意 Windows 2003 的主机防火墙及服务器前端的硬件防火墙。这里有个小技巧,你可以 telnet Windows 服务器

的 12489 端口，失败了系统会给出如下提示：

正在连接到 localhost... 不能打开主机的连接,在端口 12489:连接失败

如果连接成功了就是黑屏，这很容易区分。

2) [Setting] 项的语法跟 samba 类似，如果 allowed_hosts = 为空的话，表示客户机能被任意地址的 Nagios 服务器监控，如果此处设定 IP 的话，表示只允许此类 IP，而禁止其他 IP。

3) 注意服务器和客户端的密码不符的问题，出现上述问题应该首先检查密码是否正确无误。

4) 以前我公司的 Nagios 监控服务器置于机房内部，老是出现“could not fetch information from server”的问题，搞得外网区一片黄色，故障截图如图 1-26 所示。用 Google 和百度查了很长时间都没解决问题，后来在 Windows 客户端发现了大量的报错日志 nsclient.log: unauthorized access from 219.140.245.21，但这个 IP 根本不是我的监控服务器的 IP。后来发现这个地址是 Nagios 出去的 ADSL 地址，它在发出消息时大量阻塞。经分析发现，由于公司的服务器采用的是海蜘蛛 DMZ 映射，外接 ADSL + 铁通光纤双线的网络环境，很多时候 Nagios 服务器的地址并非固定的。后来将其迁移至电信 IDC 机房后，此问题彻底解决了。所以建议想要部署 Nagios 监控环境的读者要么选择电信，要么选择双线机房。

C:\Drive Space	WARNING	02-20-2010 13:12:14	0d 0h 15m 28s	1/3	could not fetch information from server
CPU Load	WARNING	02-20-2010 13:12:21	0d 0h 15m 21s	1/3	could not fetch information from server
Explorer	WARNING	02-20-2010 13:14:00	0d 0h 13m 42s	1/3	could not fetch information from server
Memory Usage	WARNING	02-20-2010 13:16:28	0d 0h 21m 14s	1/3	could not fetch information from server
NSClient++ Version	WARNING	02-20-2010 13:10:58	0d 0h 16m 44s	1/3	could not fetch information from server
Uptime	WARNING	02-20-2010 13:12:48	0d 0h 14m 54s	1/3	could not fetch information from server
C:\Drive Space	OK	02-20-2010 13:12:19	0d 3h 15m 23s	1/3	c: - total: 14.65 Gb - used: 11.49 Gb (78%) - free 3.16 Gb (22%)
CPU Load	OK	02-20-2010 13:15:39	0d 3h 12m 3s	1/3	CPU Load 53% (5 min average)
Explorer	OK	02-20-2010 13:15:37	0d 3h 12m 5s	1/3	Explorer EXE Running
Memory Usage	OK	02-20-2010 13:08:07	0d 3h 9m 35s	1/3	Memory usage: total 3944.00 Mb - used: 2068.48 Mb (52%) - free:

图 1-26 Nagios 的故障显示图

5) 注意语法方面的错误。很多文档都将 nsc.ini 中 [Setting] 选项的 allowed_host 写成了 allowd_hosts，这是错误的。建议用标准写法 allowed_hosts，如果有多个 Nagios 服务器 IP，之间用逗号隔开。下面摘录了 Nagios 3.0 的官方说明文档：Uncomment the 'allowed_hosts' option in the [Settings] section。

如果出现以上报错，应多关注 Windows 监控机的 nsclient.log 日志文件，从里面寻找出错的原因。另外，Nagios 服务器本身也有日志，路径名为 /usr/local/nagios/var/nagios.log。

6) Nagios 对网络的依赖还是很大的，如果我们能保证机房连接的稳定性，就可以尝试开放公网监测。但是如果不能保证，建议将其跟需要监控的机器放在同一机房进行监控。

1.6.4 用 Nagios 监控 Nginx 脚本

Nagios 默认仅支持 Apache，所以需要监控 Nginx 的话，必须在网上下载 Nginx 的相关插件——check_nginx.sh。Nginx 的监控插件脚本 check_nginx.sh 如下所示（脚本做了改动，以下配置直

接从我的监控服务器中下载，这里仅作参考。大家如果有监控 Nginx 服务器的需求，可根据如下脚本进行更改）：

```
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110 - 1301 USA
PROGNAME = 'basename $0'
VERSION = "Version 1.0,"
AUTHOR = "2009, Mike Adolphs (http://www.matejunkie.com/)"
ST_OK=0
ST_WR=1
ST_CR=2
ST_UK=3
hostname = "localhost"
port = 80
path_pid = /var/run
name_pid = "nginx.pid"
status_page = "nginx_status"
output_dir = /tmp
pid_check = 0
secure = 0
print_version() {
    echo "$VERSION $AUTHOR"
}
print_help() {
    print_version $PROGNAME $VERSION
    echo ""
    echo "$PROGNAME is a Nagios plugin to check whether nginx is running."
    echo "It also parses the nginx's status page to get requests and"
    echo "connections per second as well as requests per connection. You"
    echo "may have to alter your nginx configuration so that the plugin"
    echo "can access the server's status page."
    echo "The plugin is highly configurable for this reason. See below for"
    echo "available options."
    echo ""
    echo "$PROGNAME -H localhost -P 80 -p /var/run -n nginx.pid "
    echo "    -s nginx_status -o /tmp [-w INT] [-c INT] [-S] [-N]"
    echo ""
    echo "Options:"
    echo "    -H/--hostname)"
}
```

```

echo "    Defines the hostname.Default is: localhost"
echo " -P/ --port)"
echo "    Defines the port.Default is: 80"
echo " -p/ --path -pid)"
echo "    Path where nginx's pid file is being stored.You might need"
echo "    to alter this path according to your distribution.Default"
echo "    is: /var/run"
echo " -n/ --name_pid)"
echo "    Name of the pid file.Default is: nginx.pid"
echo " -N/ --no -pid -check)"
echo "    Turn this on, if you don't want to check for a pid file"
echo "    whether nginx is running, e.g.when you're checking a"
echo "    remote server.Default is: off"
echo " -s/ --status -page)"
echo "    Name of the server's status page defined in the location"
echo "    directive of your nginx configuration.Default is:"
echo "    nginx_status"
echo " -o/ --output -directory)"
echo "    Specifies where to write the tmp - file that the check creates."
echo "    Default is: /tmp"
echo " -S/ --secure)"
echo "    In case your server is only reachable via SSL, use this"
echo "    this switch to use HTTPS instead of HTTP.Default is: off"
echo " -w/ --warning)"
echo "    Sets a warning level for requests per second.Default is: off"
echo " -c/ --critical)"
echo "    Sets a critical level for requests per second.Default is:"
echo "        off"
exit $ST_UK
}
while test -n "$1"; do
    case "$1" in
        -help|-h)
            print_help
            exit $ST_UK
            ;;
        --version|-v)
            print_version $PROGNAME $VERSION
            exit $ST_UK
            ;;
        --hostname|-H)
            hostname = $2
            shift
            ;;
        --port|-P)
            port = $2
            shift
            ;;
        --path -pid|-p)
            path_pid = $2
    esac

```



```

        shift
        ;;
    --name -pid|-n)
        name_pid = $2
        shift
        ;;
    --no-pid-check|-N)
        pid_check = 0
        ;;
    --status -page|-s)
        status_page = $2
        shift
        ;;
    --output -directory|-o)
        output_dir = $2
        shift
        ;;
    --secure|-S)
        secure = 1
        ;;
    --warning|-w)
        warning = $2
        shift
        ;;
    --critical|-c)
        critical = $2
        shift
        ;;
    * )
        echo "Unknown argument: $1"
        print_help
        exit $ST_UK
        ;;
    esac
    shift
done
get_wcdiff() {
    if [ !-z "$warning" -a !-z "$critical" ]
    then
        wclvls=1
        if [ ${warning} -gt ${critical} ]
        then
            wcdiff=1
        fi
    elif [ !-z "$warning" -a -z "$critical" ]
    then
        wcdiff=2
    elif [ -z "$warning" -a !-z "$critical" ]
    then
        wcdiff=3
    fi
}

```

```

    fi
}
val_wcdiff() {
    if [ "$wcdiff"=1 ]
    then
        echo "Please adjust your warning/critical thresholds.The warning \
must be lower than the critical level!"
        exit $ST_UK
    elif [ "$wcdiff"=2 ]
    then
        echo "Please also set a critical value when you want to use \
warning/critical thresholds!"
        exit $ST_UK
    elif [ "$wcdiff"=3 ]
    then
        echo "Please also set a warning value when you want to use \
warning/critical thresholds!"
        exit $ST_UK
    fi
}
check_pid() {
    if [ -f "$path_pid/$name_pid" ]
    then
        retval=0
    else
        retval=1
    fi
}
get_status() {
    if [ "$secure"=1 ]
    then
        wget --no-check-certificate -q -t 3 -T 3 \
http:// ${hostname}:${port}/${status_page} -O ${output_dir}/nginx-status.1
        sleep 1
        wget --no-check-certificate -q -t 3 -T 3 \
http:// ${hostname}:${port}/${status_page} -O ${output_dir}/nginx-status.2
    else
        wget -q -t 3 -T 3 http:// ${hostname}:${port}/${status_page} \
-O ${output_dir}/nginx-status.1
        sleep 1
        wget -q -t 3 -T 3 http:// ${hostname}:${port}/${status_page} \
-O ${output_dir}/nginx-status.2
    fi
    stat_output1='stat -c %s ${output_dir}/nginx-status.1'
    stat_output2='stat -c %s ${output_dir}/nginx-status.2'
    if [ "$stat_output1"=0 -o "$stat_output2"=0 ]
    then
        echo "UNKNOWN - Local copy/copies of $status_page is empty."
        exit $ST_UK
    fi
}

```

```

}
get_vals() {
    tmp1_reqpsec='grep '^ ' ${output_dir}/nginx-status.1 |awk '{print $3}''
    tmp2_reqpsec='grep '^ ' ${output_dir}/nginx-status.2 |awk '{print $3}''
    reqpsec='expr $tmp2_reqpsec - $tmp1_reqpsec'
    tmp1_conpsec='grep '^ ' ${output_dir}/nginx-status.1 |awk '{print $2}''
    tmp2_conpsec='grep '^ ' ${output_dir}/nginx-status.2 |awk '{print $2}''
    conpsec='expr $tmp2_conpsec - $tmp1_conpsec'
    reqpcon='echo "scale=2; $reqpsec / $conpsec" |bc -l'
    if [ "$reqpcon" = ".99" ]
    then
        reqpcon="1.00"
    fi
}
do_output() {
    output="nginx is running. $reqpsec requests per second, $conpsec \
connections per second ( $reqpcon requests per connection)"
}
do_perfdata() {
    perfdata="'reqpsec' = $reqpsec 'conpsec' = $conpsec 'conpreq' = $reqpcon"
}
# Here we go!
get_wcdiff
val_wcdiff
if [ ${pid_check}=1 ]
then
    check_pid
    if [ "$retval"=1 ]
    then
        echo "There's no pid file for nginx.Is nginx running? Please \
also make sure whether your pid path and name is correct."
        exit $ST_CR
    fi
fi
get_status
get_vals
do_output
do_perfdata
if [ -n "$warning" -a -n "$critical" ]
then
    if [ "$reqpsec" -ge "$warning" -a "$reqpsec" -lt "$critical" ]
    then
        echo "WARNING - ${output} | ${perfdata}"
        exit $ST_WR
    elif [ "$reqpsec" -ge "$critical" ]
    then
        echo "CRITICAL - ${output} | ${perfdata}"
        exit $ST_CR
    else
        echo "OK - ${output} | ${perfdata}]"
    fi
fi

```

```

        exit $ST_OK
    fi
else
    echo "OK - ${output} | ${perfddata}"
    exit $ST_OK
fi

```

使用插件 `check_nginx.sh`，方法如下：

```

./check_nginx.sh -H 192.168.1.1 -P 80 -p /usr/local/webserver/nginx/ -n nginx.pid -s nginx_
status -o /tmp/ -w 15000 -c 20000

```

1.6.5 Nagios 使用心得

下面总结一下在工作中使用 Nagios 的心得：

1) 网站运维每天的工作重点之一就是监控网站的实时状态，需要时时监控。这里跟大家介绍一个监控系统主机及服务 Nagios 系统实时提醒的 Firefox 插件，很实用。如果上班时需要实时关注监控服务器主机及服务的 Nagios 系统，就得一直开着一个网页，然后让页面自动刷新，这样有点儿麻烦，而且浪费资源。虽说有一款 Nagios 辅助小软件比较方便，可以将其最小化到任务栏，有异常时会出现浮动提示窗口，但是我一向喜欢尽量将所有东西都嵌入浏览器中进行操作，所以更希望能找到一款 Firefox 插件来实现类似于此辅助软件的功能。后来终于找到了，这个插件即 Nagios Check 插件。它的安装方法非常简单，这里就不多费篇幅了。效果如图 1-27 所示。



图 1-27 Nagios Check 嵌入 Firefox 浏览器图示

2) 业务网站最好置于自己的机房内，因为 Nagios 对网络的依赖性很大。它依靠 ping 来检测服务器是否存活，如果网络情况不好或因别的原因造成 Nagios 检测不到监测服务器的话，会造成一个啼笑皆非的后果，就是它会狂报警，说此服务器已宕机，属于 Critical 情况。但事实上此服务器情况良好，仅仅是跟 Nagios 机器的网络不通而已。如果遇到这种情况希望大家能甄别。

3) 由于 Nagios 是部署在内网中的，所以它只能对内网的所有机器进行监控。由于我们的网站都是将内网机器的 IP 映射到防火墙的公网地址，对此 Nagios 就无能为力了。事实上，防火墙的性能和稳定对网站的影响也是很大的，这个时候我们可以购买类似于 AlertBot 的实时扫描服务器来扫描我们的商务网站，并配合 Nagios 对网站进行实时监控。如果是防火墙或机房出问题了，它会立即向你发送警告邮件。如果你的邮箱同时收到 Alertbot 和 Nagios 的报警，一定要慎重对待，这种情况百分百就是内网机器出了故障。

4) 对于公网中的一些重要机器，我们可以将 Nagios 部署在稳定的电信或双线机房中监控。

5) 如果嫌在 Nagios 下部署短信猫或飞信麻烦, 可以尝试一下移动免费的 139 邮件短信业务。大家应该也明白, 并不是每家公司都愿意在这些方面投入大量资金, 很多公司都是抱着能省则省的原则。

6) 有时我们的系统组会有这个需求, 希望在系统繁忙时能留下日志, 以便分析到底是受到了攻击, 还是开发人员设置不当, 抑或是运维人员改动了系统配置等。机器少时可能问题不大, 如果公司的 CDN 服务器集群是一百多台, 而且还在增长, 那么就可以使用基于 vmstat 的 SHELL 脚本来做为 Nagios 的补充, 让其在系统繁忙时分离出日志, 供大家分析问题, 找出问题的症结所在。

7) 如果有某台服务器的某一项 (比如我们的 jail), 本来就是 8 台虚拟机一起工作于线上环境的, 负载很大, 但 Nagios 就是不停地报警, 认为这不正常, 这时可以直接点中此服务器的负载项, 然后选择 Disable notifications for this service, 然后这世界就清静了。我们直接将 FreeBSD 的 jail 虚拟机用于线上环境, 它配置起来很方便和高效, 但也有个缺点, 由于下面的子机全都是共用原宿机的 CPU、内存及磁盘, 所以任何一台子 jail 机的负载过高或使用了磁盘空间, 都会导致原宿机的 Nagios 报警。目前解决的办法是, 尽量将磁盘 I/O 和 CPU 占用高的服务单独分离出去。

8) Dell 系列的服务器在 RAID 充电时也有类似情况, 我们也可以按照以上办法处理。

本节重点介绍了 Nagios 这款监控工具, 同时也分享了一些个人用此软件的经验心得。由于线上环境对高可用要求近乎苛刻, 所以我们必须随时掌握服务器的性能, 及时根据 Nagios 的警报邮件或短信来处理服务器的故障。

1.7 项目实施中应该注意的事项

项目施工工程师可能会实施不少公司外包的项目, 像并发量大的 CDN 广告网站, 还有证券类资讯网站, 以及电子商务网站等, 还有就是一些并发量不多的 Linux 集群广告网站等。个人感觉其实在项目实施的过程中还是有很多技巧和心得的, 而且项目实践的锻炼也会让人成长得很快。下面将介绍一下在项目实施过程中我们应该注意哪些事项, 希望能引起大家 (尤其是项目实施工程师们) 的重视。

1) 准备工作一定要做好做细。项目实施属于比较有技术含量的工作, 有时还会因为合同期限会对时间有限制, 所以一定要在公司将所要部署的施工方案细化, 最好是做个几十遍, 将各种各样的意外都设想进去。众所周知, Linux 下的软件如果是源码编译安装的话, 有的也区分 32 位和 64 位, 更别说 Windows 下的程序了。在吃了一次亏后, 我就直接建议, 以后程序的开发环境和测试都统一为 64 位, 部署环境则都统一为 64 位的 Centos5.x 及 Windows 2003 企业版 + SQL Server 2008。这说的是服务器操作系统的统一性。再说一下软件版本的问题, 比如说我们要用 Nginx 作负载均衡, 这时候是用 Nginx 0.7x 还是 Nginx 0.8.15 呢? 经过大量测试, 包括相关的压力测试, 最后决定用 Nginx 0.8.15, 因为其稳定性相当不错。所以后来的项目实施中一直采用 Nginx 0.8.15, 其他的软件版本也会进行类似测试, 测试的工作一定要做好做细多做, 虽然可能会占用不少时间和精力, 但这项工作是必需的。另外就是要跟客户的 IT 技术部门多沟通, 弄清楚他们的机房带宽, 服务器是放在自己机房还是放在电信机房, 值班人员是否 24 小时监控, 服务器的性能如何, 是否是四核至强, 单台大约能承受多少并发量 (很遗憾, 这个问题基本没多少 IT 技术人员能够很清楚), 最好的做法是给对方的 IT 技术人员一张关于网络和系统的调查表, 然后一定要得到对方肯定的反馈。我最喜欢的情况之一是对方什么都没有, 从路由器到服务器到存储都是由我们推荐, 由他们购买, 这种情况就能对服务器的性能一清二楚了。根据我所了解的情况, 系统管理员一般都很熟悉 Windows 系列, 但一谈到 Linux 和 Unix, 尤其是 Centos 和 FreeBSD, 基本就不太明白了, 所以一定要耐心跟他

们核对。另外有一件事需要提醒大家，我们带过去的程序，如果是 Linux 的源码和数据库文件的话估计还好，但如果是 Windows 文件，尤其是可执行文件（一般是用移动硬盘直接携带），一定要用最新的杀毒软件杀毒，我们一般用 NOD32 和卡巴斯基，但这还不保险，最好是跟对方的技术支持人员多沟通，用对方的杀毒软件来进行检查。在这一点上我们是有教训的，如果你的软件有病毒，损失会全由己方来承担，而且这对于系统管理员来说就太不专业了。我负责的多是 Linux/Unix 机器（我安装的 MySQL 机器也很多），所以我这里基本没出过类似问题；而同事携带的移动硬盘，由于用过的人太多，结果拿到客户的机房一扫描，居然发现有病毒。所以这事最好是在己方公司做好，不要因为细节影响大局。另外要多了解一下对方是否有网络安全方面的防火墙，一定要有一台防 DDoS 攻击的。目前我们的客户多是做证券系统和电子商务的，非常注意安全，一般都有 Juniper 系列的硬件防火墙。

2) 在项目实施过程一定要注意多部门的沟通和配合。这一点其实也很重要，一套系统里一般包含着程序、数据、图片等文件，所以作为项目实施人员，平时一定要注意与各部门多沟通多了解，必要的时候可以跟开发人员一起工作和探讨问题。其实作为系统工程师，即 System Admin，你可以不精通但一定要懂这些语言，比如 PHP、Java、C++ 等。另外跟数据部门也要多注意交流，了解数据库是运行在什么平台下的，是 MySQL 还是 SQL Server 2008，这样才能有的放矢、胸有成竹。公司内部交流基本还算比较顺利的，难点就在于项目施工过程中的多部门协调和交流。我们会提前跟客户的业务领导打招呼，约定在一个时间段将所有的施工人员召集起来开一个交流会。比方说，我们要在南京做某证券系统，我们从武汉出发，3 小时的动车到南京。下了火车后直接去客户的办公室，拿出了项目的施工拓扑图，迅速地跟所有的施工人员开了一个会，说明了项目施工中的重点和困难。虽然这样做确实很辛苦很累，但这事最好提前做，并且要迅速，谁都不知道在项目实施过程中会出现什么样的意外。第二天我们去电信机房时，发现 HP 的工作人员一下子来了 6 位，刚开始我不是特别理解，后来明白了，原因就是 4U 的服务器确实长得“很彪悍”，人少了动不了它。还好 HP 想得比较周到，派的都是年轻力壮的小伙。那个项目属于中等项目，一共有 6 位 HP 的工程师，华赛派了一位网络工程师，天泰派了一位安全工程师，我们的客户是派了 3 个 IT 技术支持过来，我们这边则是我和同事两人。人多力量大，各司其职，迅速做事，效率还是很高的。HP 用 KVM 安装系统，8 台服务器同时安装。另外的两位工程师也很迅速地设定 IP，做内网 IP 映射等。最后，我担心的意外都没发生，一天之内将所有服务器的系统都正常安装了，同时开放了 Linux 的远程（SSH：22）和 Windows 的远程（我们是用 Remote Admin），网络方面也基本畅通。天泰的安全工程师调试了一天，就开启了防火墙的透明（Transparent 模式）。这是比较成功的案例，失败的例子我也经常遇到，比如网络不通或对方将我们的网络设置错误等，那就比较折磨人了，希望大家在实施时注意。

3) 迅速和稳定地处理项目实施中遇到的突发事件。这一点就看项目实施人员的经验和能力了。比如我有次去武汉某证券公司部署 Nginx + Apache 的“1+3”的小型网站时，忽然发现有台 HP360 的主板坏了，怎么也装不上系统。我只好放弃这台，只部署了“1+2”的方案，保证系统还是按约定的时间上线。然后我又跟公司的 Qs 做了大量的压力测试，发现“1+2”的方案能顶上，所以就暂定用它了。还有一次，我们去实施时，发现客户已请了家外包公司将机房的网络做好了。由于客户要求做 HA 系统，所以我们在施工拓扑图上多画了一个网段，即 10.0.0.0 网段，然而客户这边的 IT 老总将意图理解错了，居然将整套系统做成了两个网段，还划分了 VLAN。原本只是局域网中的一套系统，怎么搞得这么复杂啊？没办法，只好花了一下午的时间跟他解释什么叫 Heartbeat，什么

叫双绞线。等他明白过来，再让外包公司的人过来重新布线和分配交换机的 IP 时，宝贵的工期已足足过了 3 天。这 3 天我只能先将系统和能部署的全部署上去，最后只等网络 ping 通了。期间也想过增加一条静态路由的方法，但对方的网络工程师不停地做抽插网线的实验，最后只有作罢。还有一件事，那是替客户将所有系统部署好后，发现 80 端口怎么也通不了。仔细问了对方的网络工程师，确定问题不是出在对方那里（熟练的网工难道还不会映射一个 80 端口？），我用远程 Telnet 对方映射的外网 IP 80 端口时发现毫无反应，忽然想到：是不是机房封了 80？难道这么简单的问题事先没沟通好？立即让对方的老总跟机房备案开通了 80 端口，问题就解决了。总之，在项目施工中，会不断遇到许多不顺利的事，让你焦头烂额，这就要看你的应变能力了。注意，这时候耐心地跟对方沟通很重要，毕竟人跟人之间需要交流。你要理解的是：你可能是某方面的权威，但在另一方面，你可能什么都不算。所以，低调做人、高调做事，这也是项目施工中应该注意的。

4) 在项目施工中多学习，提高个人的能力。实际的线上环境还是很锻炼人的。我个人感觉，一个成熟且安全的系统，还是很有技术含量的。你要了解网络、程序、数据库，还要了解什么是并发，什么是 SSL（哪家公司支持多域名的 SSL），什么叫网架架构。你的网架安全吗？监控怎么办呢？文件服务器压力大吗？单台 Web 崩溃了怎么办？PHP 的 Session 同步怎么解决？MySQL 数据库做成什么样的？是主从复制还是做成 MySQL-cluster（个人认为主从复制更适用于线上系统）？你的 SQL Server 2008 是单台还是做成 SQL Server 2008 故障转移群集？shell 很熟悉了吧，Apache 下的正则熟悉吗？能不能在 10 分钟内搞定？有时候，做系统集成挺累的，你要掌握的东西太多了，而网工相对而言压力就小多了，你只要保证网络通畅即可。不知道大家有没有遇到这种情况，老板经常会站在身后，看着你处理故障，所以如果 2 分钟能处理的事，我绝对不会拖到第 3 分钟。没办法，压力逼着人成长。项目实施一般都是有合同时间期限的，所以需要大家在短期内做好部署工作，测试这块可以跟对方商量，对方也要花一段时间进行内测的。完成一个成熟安全的系统后，你会发现自己成长得非常快，平时不太注意的细节在线上就是重大的安全生产问题了。所以只要公司有外出部署的项目，我一般都会参加，毕竟可以有机会接触到各种各样的服务器及小机，还有防火墙等。

5) 系统的测试和监控。压力测试和性能测试一般由公司和对方的测试人员进行，这一般都会给比较充分的时间，特别是压力测试这块，需要耐心地和对方的 QS 沟通，顺便熟悉他们用的 Load Runner（这个软件的使用还是很复杂的），或者是 Apache 自带的 ab 工具和 webbench 等。监控这块我强烈建议用 Nagios，邮件和短信通知都要做，并要求对方的系统工程师 24 小时开机，遇到紧急事件要即时处理。事实证明，如果都是做成双机 HA，并且有硬件防火墙的话，紧急事件还是比较少的。

以上是我在项目实施工程中的一些经验和心得，与大家分享一下。记住，技术不能完全左右项目的成功与质量，因为技术只占其中一半的分量，有可能一半还不到。成功实施一个成熟且安全的网站或系统，其中的成就感和酸甜苦辣也只有自己知道，希望大家能从中得到帮助（特别是项目实施人员和系统管理员）。

1.8 小结

本章从 Centos5.5 x86_64 的安装、网络配置、日志分析、性能及状态监控、优化及虚拟化等方面对 Linux 硬件进行了全方位的说明，这些都是构建高性能及高可用 Linux 系统的基础，希望大家能够掌握此章内容。尤其是 Linux 服务器的网络配置相关内容对于我们以后的工作会有很大的帮助。理解了本章内容，我们的工作会更加得心应手。



第 2 章

FreeBSD8.1 在企业中的部署应用

- 2.1 最小化安装 FreeBSD8.1
- 2.2 最小化安装 FreeBSD8.1 后的升级优化部署
- 2.3 在 FreeBSD8.1 下部署 jail 虚拟机
- 2.4 在 FreeBSD8.1 下搭建版本控制服务器
- 2.5 在 FreeBSD8.1 下搭建 Samba 文件服务器
- 2.6 在 FreeBSD8.1 下配置 NFS 文件服务器
- 2.7 在 FreeBSD8.1 与 Centos5.5 下搭建 rsync 服务器
- 2.8 在 FreeBSD8.1 下搭建 vsftpd 服务器
- 2.9 在 FreeBSD8.1 和 Centos5.5 下搭建 PHP 与 Java 应用环境
- 2.10 小结

在工作中我们经常会听到“开发环境”这个词汇，那么在实际工作中，我们该如何快速搭建大量的开发环境和测试环境呢？通过对 Centos5.5 x86_64 与 FreeBSD8.1 x86_64（以下统一简称为 FreeBSD8.1）的比较，我们发现在开发环境中，FreeBSD8.1 在稳定性与实用性方面有自己独有的优势，特别是它的 jail 虚拟机，通过 ezjail 工具，可以快速部署上百套开发和测试环境。本章将会详细讲解 FreeBSD8.1 的安装、最小化安装后的优化、jail 虚拟机的部署，以及 FreeBSD8.1 下的应用部署。希望大家通过本章的学习，能够很好地掌握 FreeBSD8.1 的基础知识，并且可以快速地搭建自己的开发环境和测试环境。

2.1 最小化安装 FreeBSD8.1

先说明一下：为了截图方便，本书选择了在 VMware Server 上安装系统（在服务器上通过光盘安装 64bit FreeBSD8.1 的过程与此大同小异），虚拟机的硬盘设定为 10GB，内存设为 512MB，分区情况如下：

```
/ 2048M
swap 888M
/usr 7300M
```

无/boot 分区，如果选择/boot 分区，则需要手动引导系统，所以这里就不创建/boot 分区了，这也是与 Centos5.5 等 Linux 操作系统不一样的地方，注意不要混淆。

注意 FreeBSD 系统对于许多人（尤其是熟悉了 Windows 和 Linux 操作的人）来说，比较陌生。但无可否认的是，FreeBSD 是一个优秀的系统，我希望大家能从最基础的安装和配置开始慢慢熟悉它、了解它，相信你最后会喜欢上它的。

如果最小化安装 FreeBSD8.1，则只需要 FreeBSD8.1 的第一张光盘，它的下载地址为 <ftp://ftp.freebsd.org/pub/FreeBSD/releases/amd64/ISO-IMAGES/8.1/FreeBSD-8.1-RELEASE-amd64cd1.iso>，用迅雷下载即可。下面将详细讲解安装的过程与步骤。

（1）启动 VMware Server，进入到 FreeBSD8.1 的安装界面，如图 2-1 所示。我们选择其中的第 1 项（默认项）。



图 2-1 FreeBSD8.1 安装界面

(2) 选择使用区域，我们可以根据实际需求来选择“United States（美国）”或“China（中国）”，如图 2-2 所示。

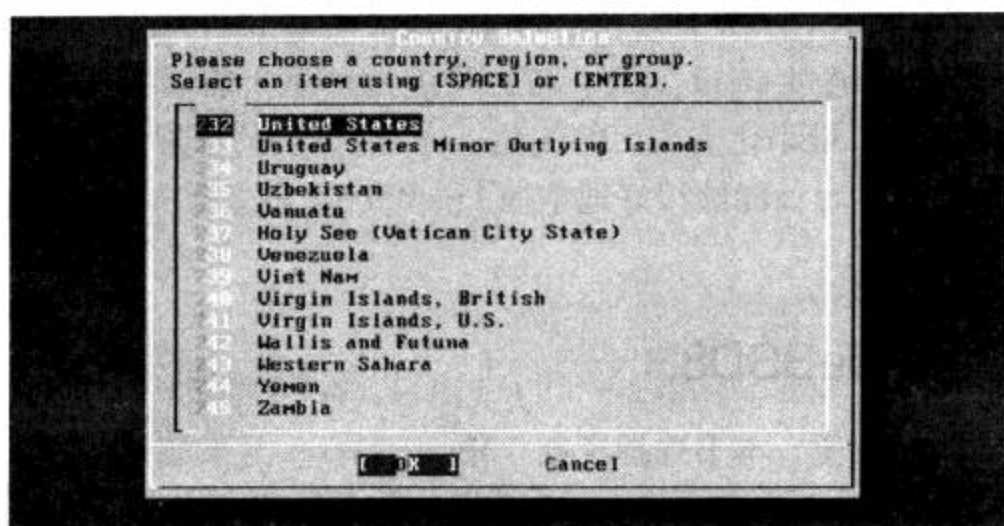


图 2-2 选择使用区域

(3) 选择键盘的制式，我们选择“USA ISO（美式）”键盘，如图 2-3 所示。

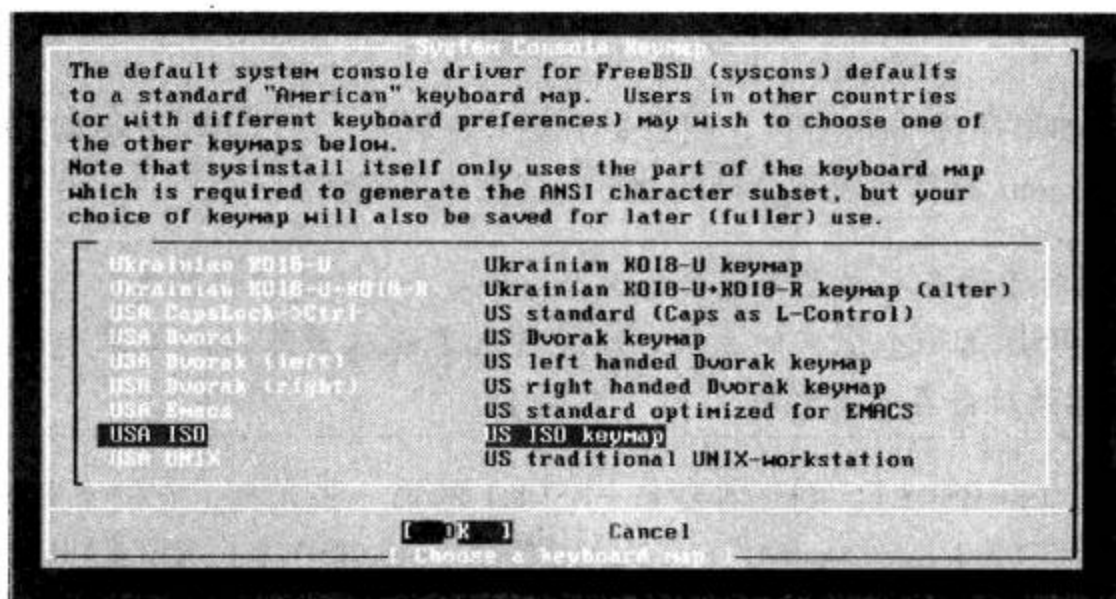


图 2-3 选择键盘制式

(4) 选择“Standard（标准）安装方式”，如图 2-4 所示。

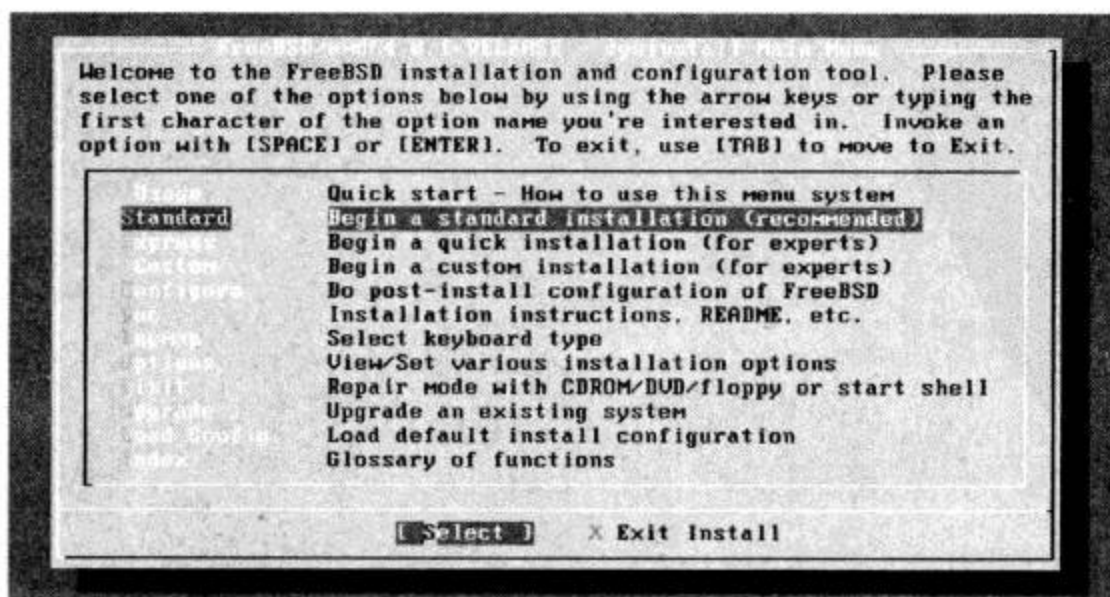


图 2-4 标准安装方式

(5) 系统进入了硬盘划分界面, 这时系统会询问是否要将整块硬盘划分给 FreeBSD8.1 使用, 为了工作方便, 通常我们会将服务器的整块硬盘 (无论是做了 RAID1, 还是 RAID5, 或其他 RAID 模式的整块硬盘) 都划分给 FreeBSD8.1 使用, 即选择 “A” 和 “Q”, 如图 2-5 和图 2-6 所示。



图 2-5 选择 “A” 是使用整块硬盘



图 2-6 选择 “Q” 完成硬盘选择过程

(6) 系统让我们选择以何种方式安装启动管理器, 我们选择第一项, 即标准方式 (不需要互动), 这也是系统默认推荐的方式。其他几项中, 第二项 “BootMgr” 表示如果你的系统中存在着其他系统, 在选择此方式安装后可以选择进入哪个系统, 第三项 “None” 表示保留已存在的 boot manager, 如图 2-7 所示。

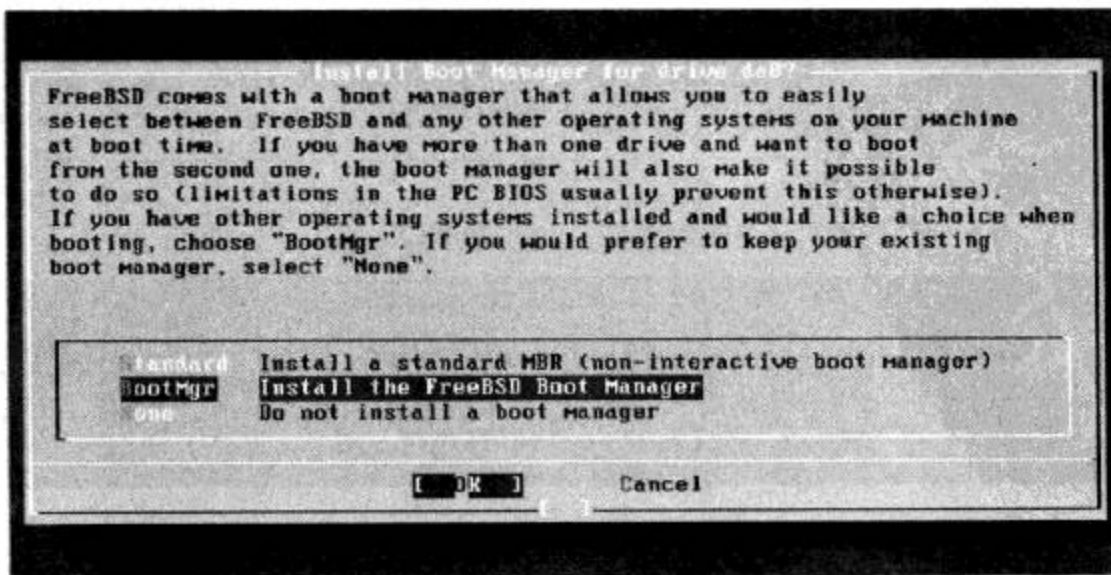


图 2-7 选择标准引导方式

(7) 系统提示我们建立 FreeBSD 分区，如图 2-8 所示。

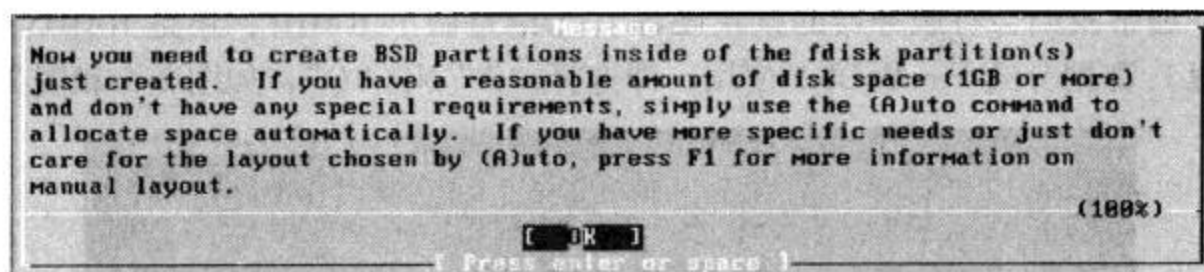


图 2-8 系统提示建立 BSD 分区

(8) 系统进入分区界面，FreeBSD8.1 提供全自动分区方案，我们选择“A”和“Q”即可完成此分区步骤。但是在工作中并不推荐这样做，建议了解 FreeBSD8.1 分区的详细过程，并根据需求来设定所有分区的大小。首先，我们建立大小为 2048MB 的/分区，然后进入 FreeBSD 的分区选项界面，如图 2-9 所示。

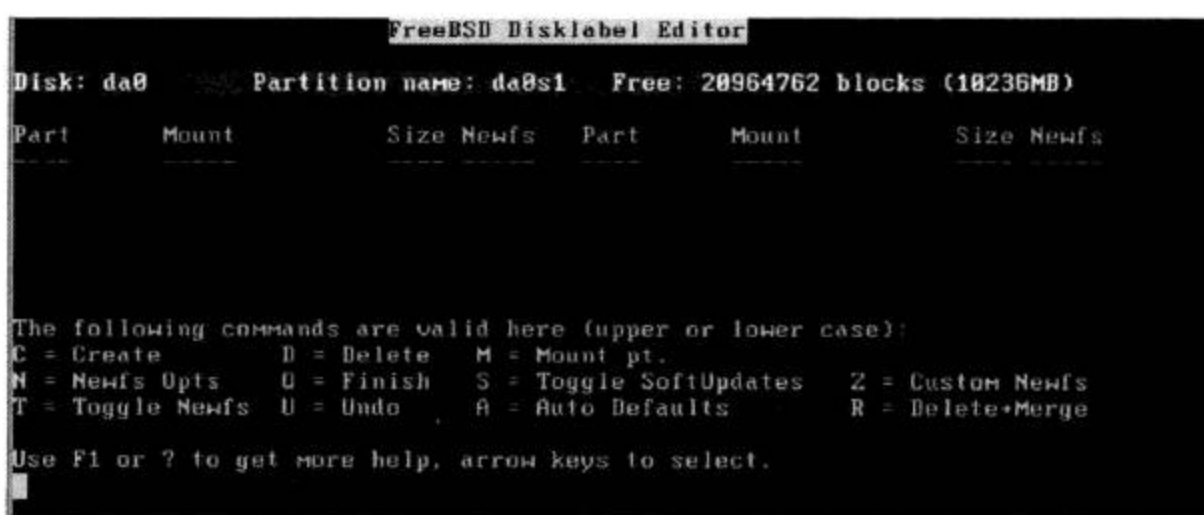


图 2-9 FreeBSD8.1 的分区界面

(9) 在图 2-10 中，我们选择“C”，然后输入“2048M”。

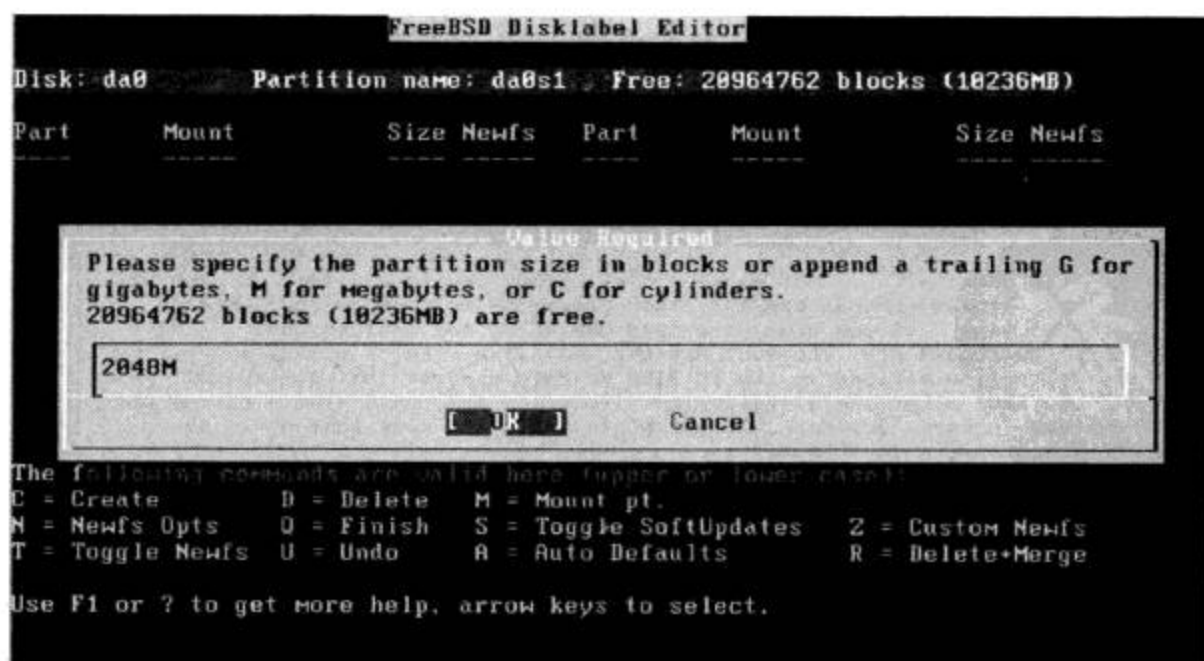


图 2-10 选择/分区的大小

(10) 当要选择/分区的文件类型时，我们选择 FS 类型，如图 2-11 所示。

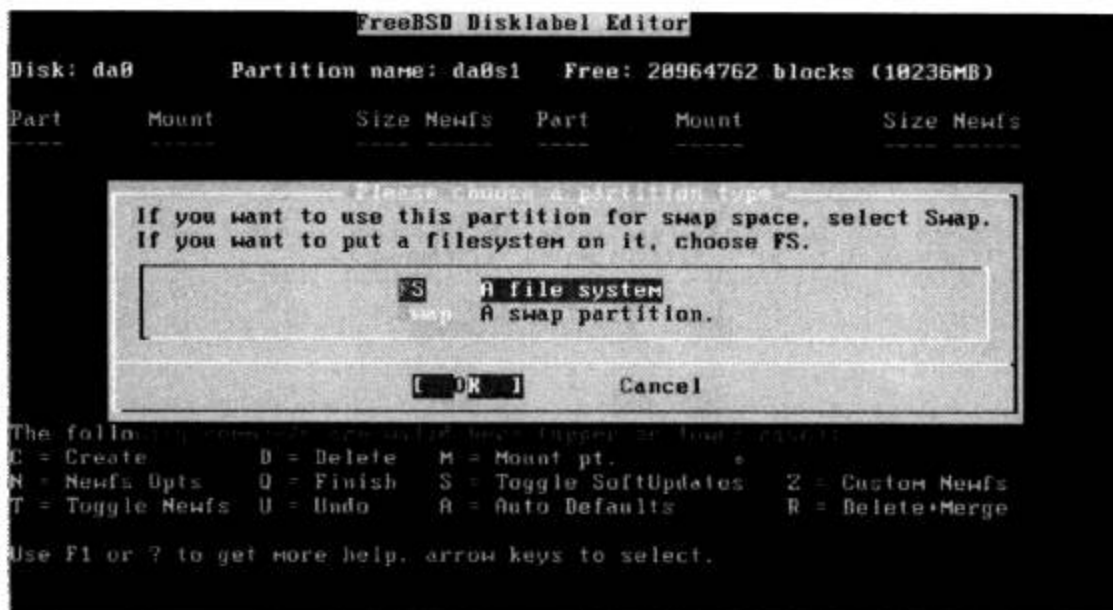


图 2-11 选择 FS 作为/分区的文件类型

(11) 关于挂载点，我们选择/，如图 2-12 所示。

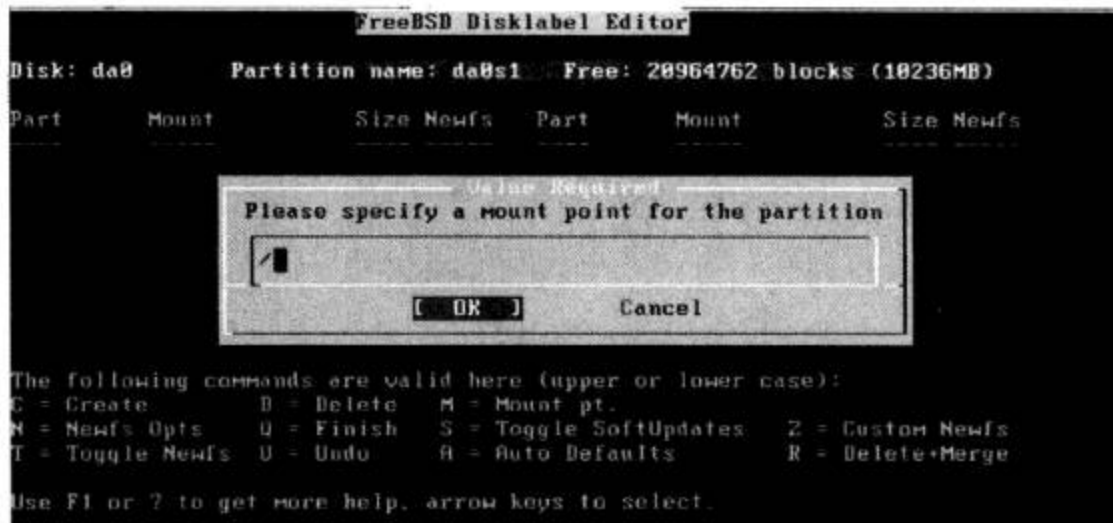


图 2-12 选择/分区的挂载点

(12) 接下来划分 swap 分区，由于本例中内存大小为 512MB，所以 swap 分区的大小可以选择 512 的 1.5~2 倍，本书选择的是 888MB。如果在服务器上安装，内存大小为 4GB，则可以选择 6GB 左右的空间给 swap 分区，其他大小以此类推。划分 swap 分区的步骤如图 2-13 和图 2-14 所示。

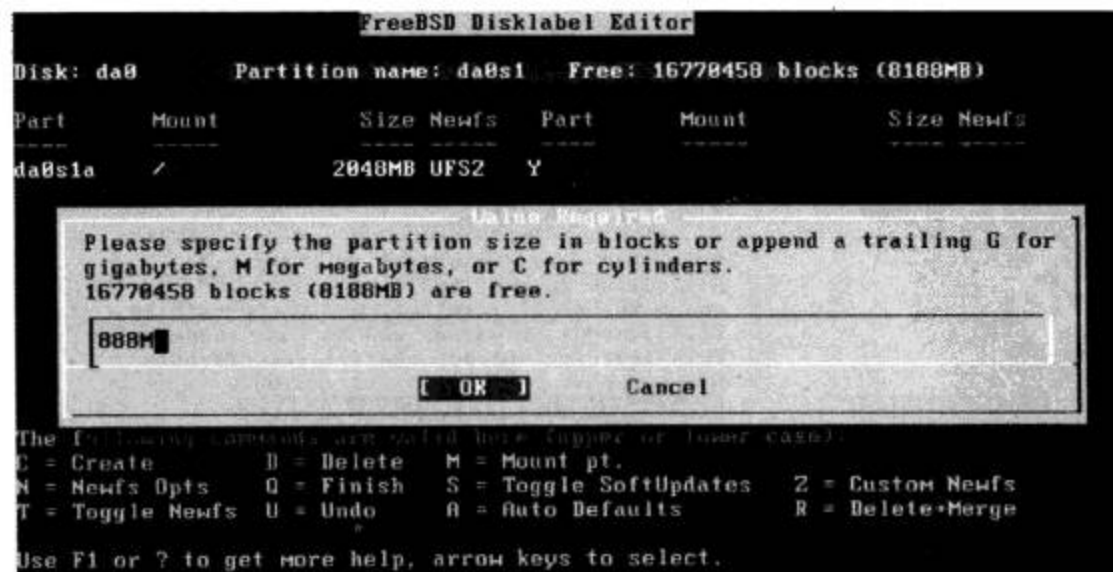


图 2-13 选择 swap 分区的大小

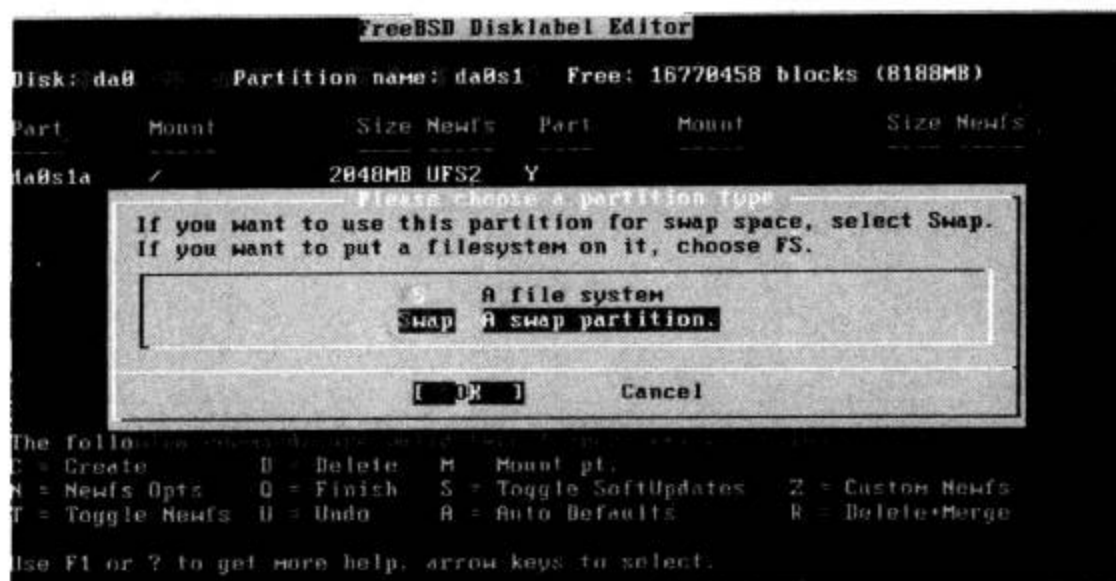


图 2-14 选择 swap 分区的文件类型

(13) 下面创建/usr分区，步骤与 swap 分区类似。安装完成后，此例中的 FreeBSD8.1 分区如图 2-15所示。

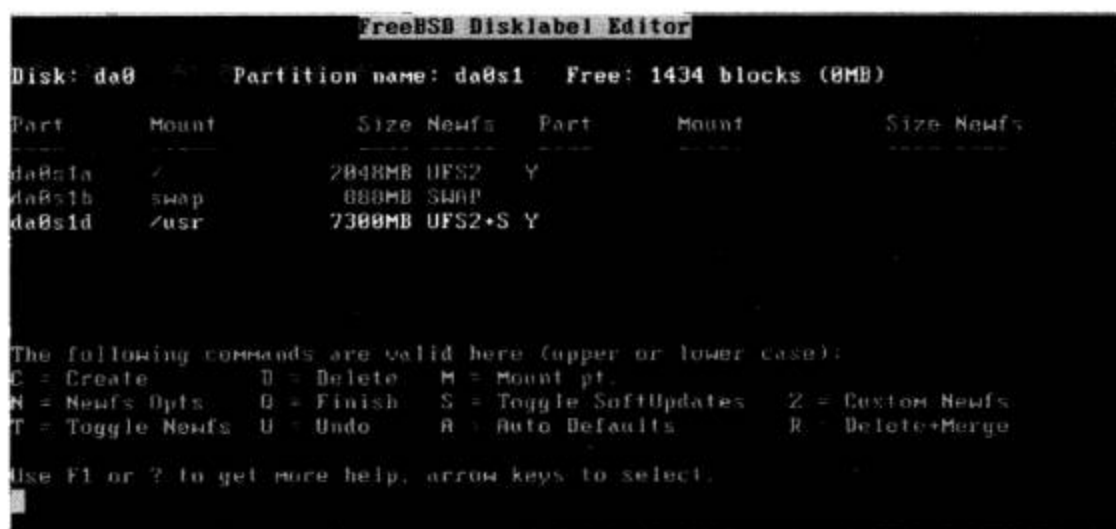


图 2-15 本例中的 FreeBSD8.1 的分区情况

(14) 系统让我们选择采用哪种安装系统的方式，我们选择最小化安装，如图 2-16 所示。

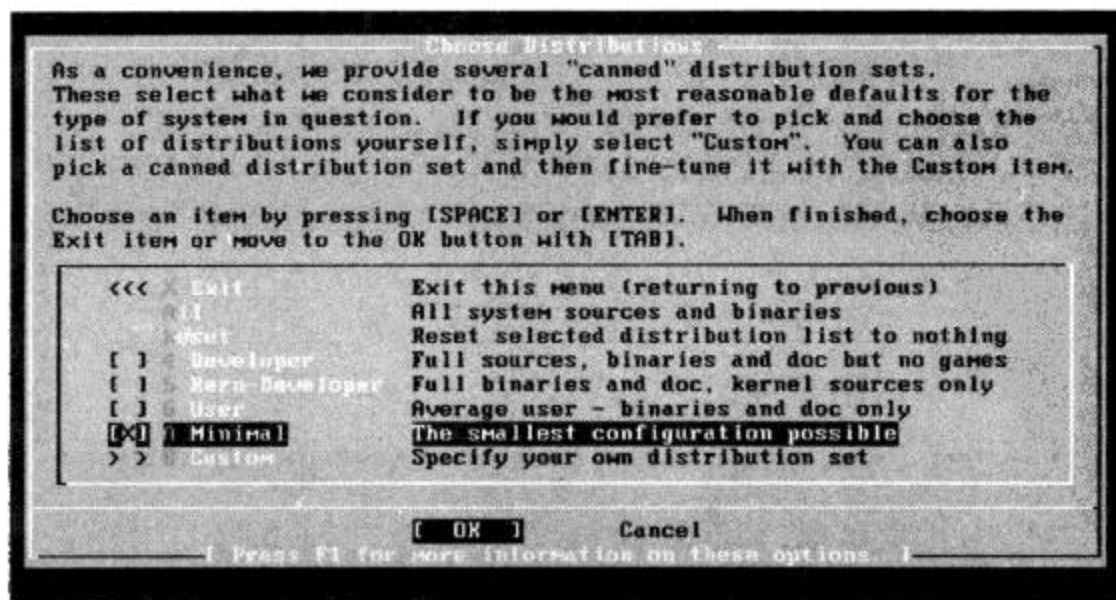


图 2-16 选择安装与系统的方式

(15) 系统提示从哪种媒质中安装 FreeBSD8.1, 我们选择以 CD/DVD 的方式安装, 如图 2-17 所示。

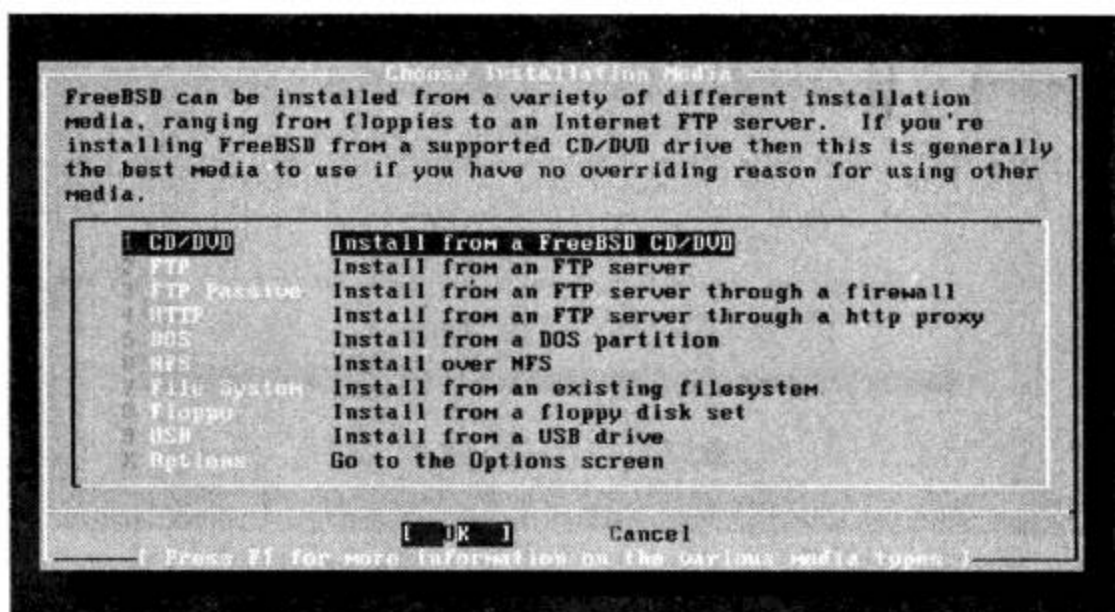


图 2-17 选择以哪种媒质安装 FreeBSD8.1

(16) 这时, 系统会提示马上要进行格式化和安装系统了, 是否选择继续。这是最后改变的机会, 我们选择 “Yes”, 如图 2-18 所示。

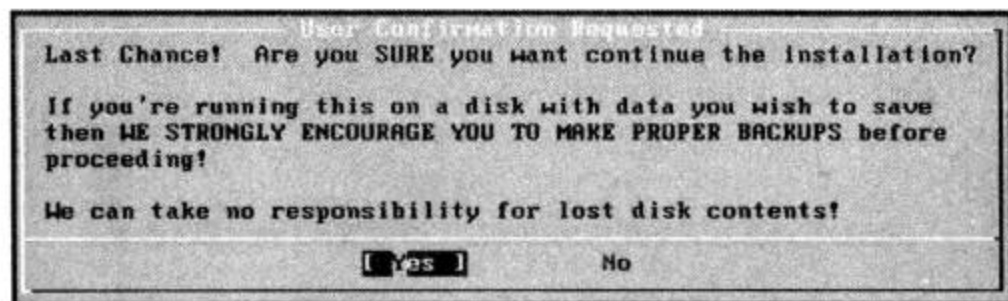


图 2-18 系统提示是否继续安装

(17) 安装结束后, 系统会提示用 sysinstall 来配置系统, 如图 2-19 所示。

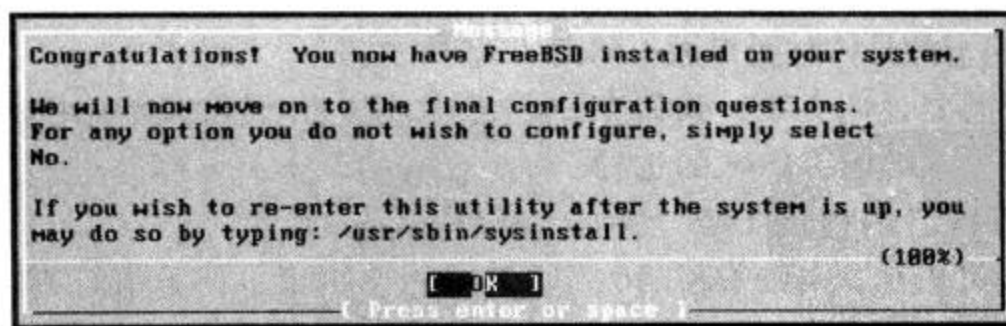


图 2-19 系统用 sysinstall 工具配置系统

(18) 接着系统会询问是否需要配置以太网卡, 我们选择 “Yes”, 如图 2-20 所示。

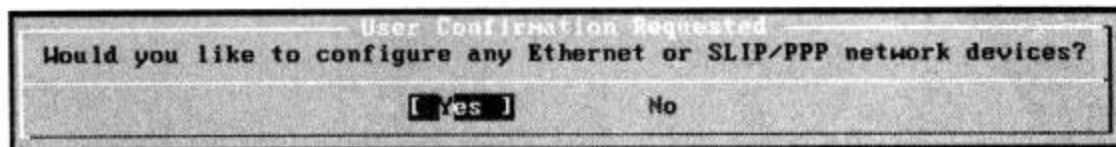


图 2-20 系统提示是否要配置以太网卡

(19) 在接下来的过程中, FreeBSD8.1 会为我们配置网卡 (默认的为 em0), 由于现在服务器都在防火墙后面或置于 IDC 机房中, 所以我们不需要进行 PPP 拨号。这时系统会询问我们是否需要进行 IPv6 的配置, 选择 “No”, 如图 2-21 所示。

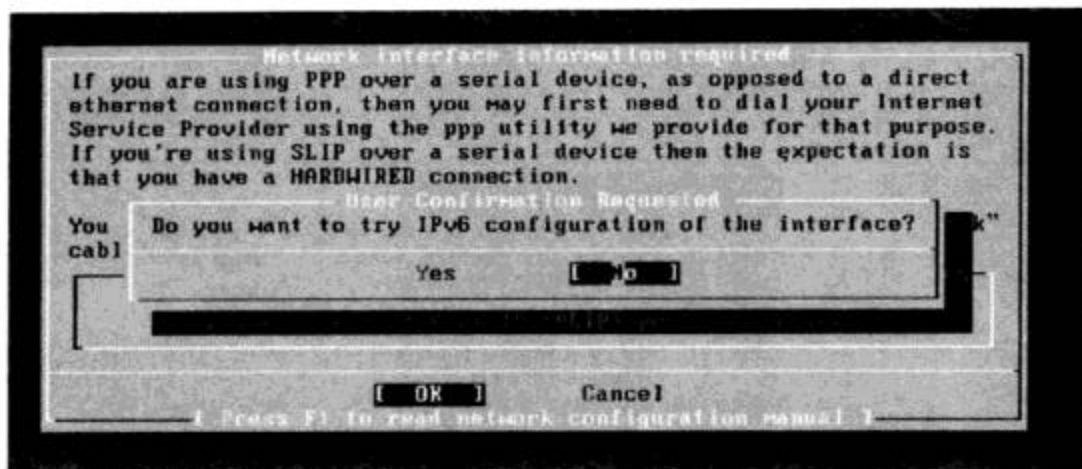


图 2-21 不需要此网卡配置 IPv6

(20) 系统会继续问我们是否需要用 DHCP 服务器自动分配 IP, 在这里我们想自己配置, 所以选择 “No”, 如图 2-22 所示。

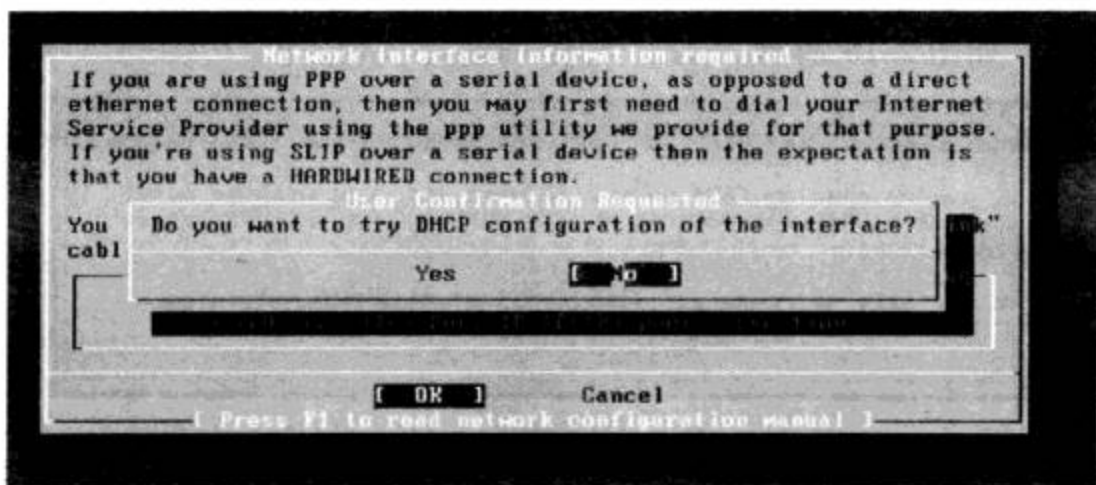


图 2-22 选择是否配置 DHCP

(21) 系统进入网卡的具体配置界面, 如图 2-23 所示。

这里有一个情况需要说明一下, 我们在为本机配置 FreeBSD8.1 系统时, DNS 的选择有多种情况, 如果有自己的内网 DNS 且为 192.168.1.10, 这里就可以填入 192.168.1.10; 如果是公网 IP, 则填入公网地址, 如 203.103.96.112; 如果是经过路由器 NAT 出去的, 则直接填入路由器的地址, 即 192.168.1.1。当然, 因为实际工作环境的最前端都有防火墙, 所有后端的机器都是通过路由器 NAT 经防火墙上网的, 所以我们直接填写公网地址 (如 203.103.96.112 或 8.8.8.8) 即可, 内网的

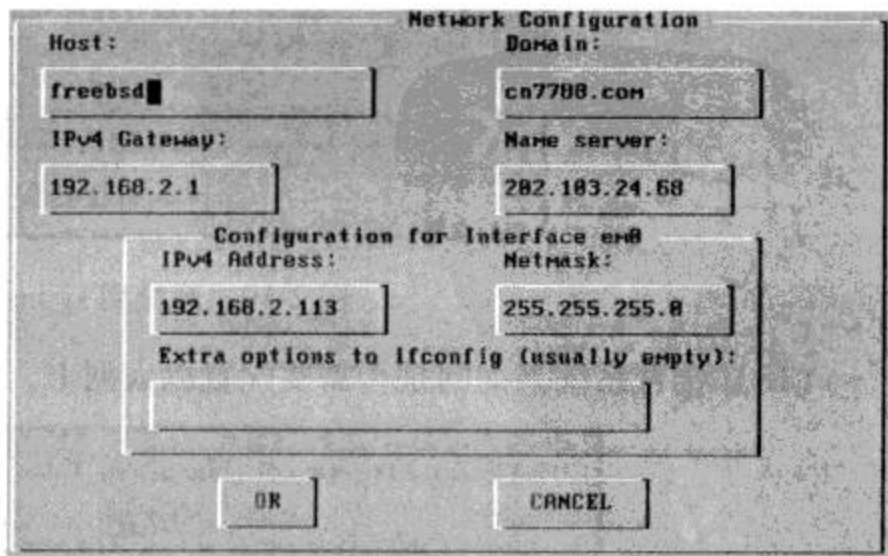


图 2-23 网卡配置

DNS 地址也可。每个人的上网环境不一样，这里可以根据环境变化而变。

注意 Extra options to ifconfig 是你将要添加的一个特殊的接口选项。在这个例子中没有，所以我们选择为空。

(22) 系统提示是否立即启动 em0 网卡，选择 “Yes”，如图 2-24 所示。

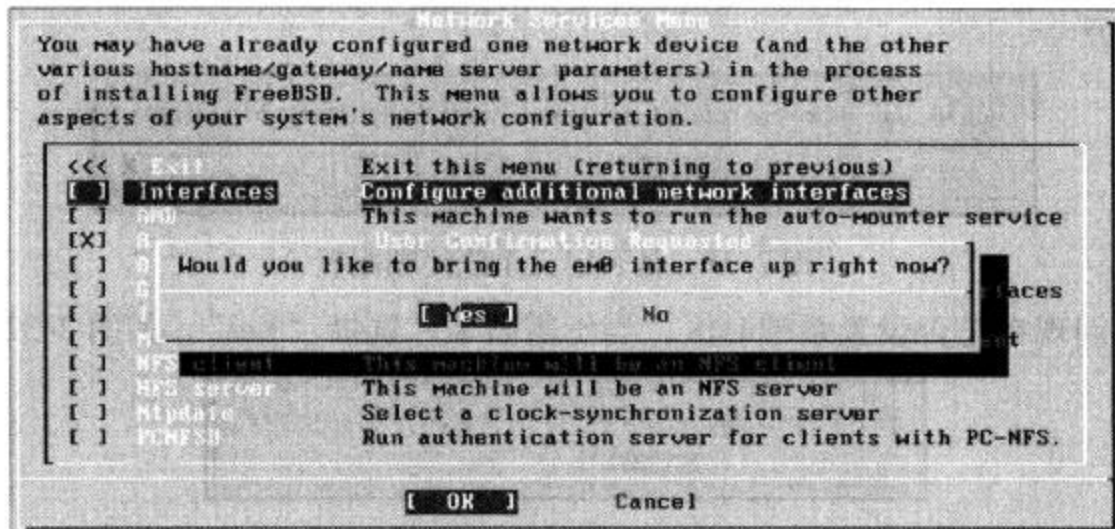


图 2-24 是否立即启动 em0 网卡

(23) 系统提示是否将此机器作为一个网关，选择 “No”，如图 2-25 所示。

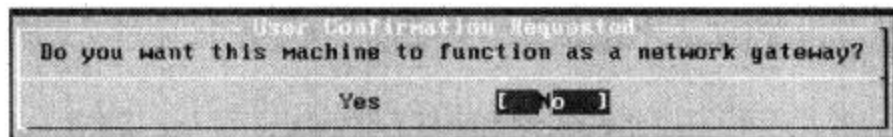


图 2-25 是否选择将此机器作为网关机器

(24) 系统询问是否需要配置 inetd 和其他的网络服务，选择 “No”，如图 2-26 所示。

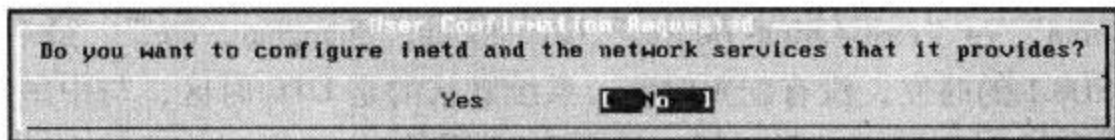


图 2-26 是否配置 inetd 和其他网络服务

(25) 系统询问是否需要 SSH 登录，即开启 OpenSSH 服务，如果开启，系统会打开 22 端口，允许安全地登录系统，这里选择 “Yes”，如图 2-27 所示。

(26) 系统询问是否允许匿名用户 FTP 进入此机器，选择 “No”，如图 2-28 所示。

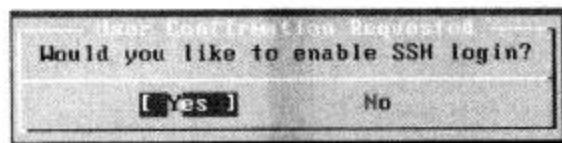


图 2-27 是否需要 SSH 登录

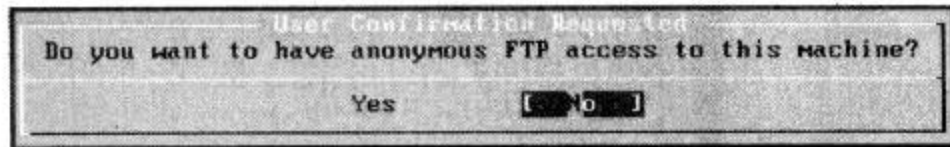


图 2-28 是否允许匿名用户 FTP 进入此机

(27) 系统询问是否将此机器配置成 NFS 服务器或客户端，选择 “No”，如图 2-29 和图 2-30 所示。

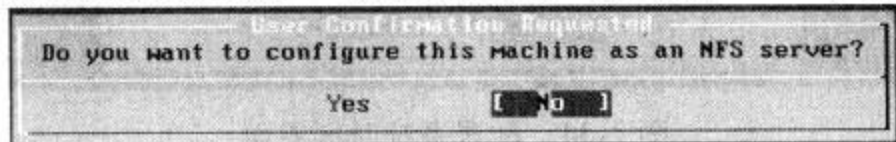


图 2-29 是否将此机器作为 NFS 服务器

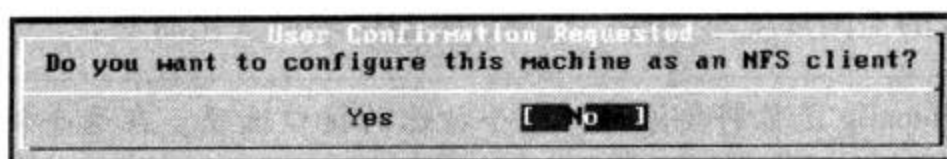


图 2-30 是否将此机器作为 NFS 客户端

(28) 系统询问是否定义系统终端配置，如字体大小、光标移动速度等，可根据具体环境来决定，如图 2-31 所示。

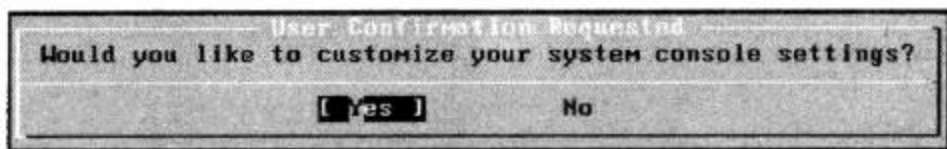


图 2-31 是否定义系统终端配置

(29) 系统询问现在是否设置系统时区，当然需要了，选择“Yes”，如图 2-32 所示。

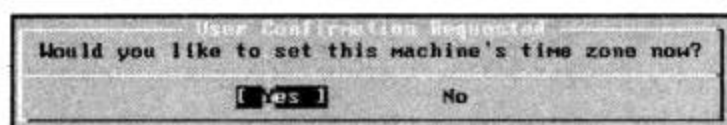


图 2-32 是否设置系统时区

(30) 需要将系统的 CMOS 时钟定义为 UTC 制式吗？这里选择“Yes”，如图 2-33 所示。

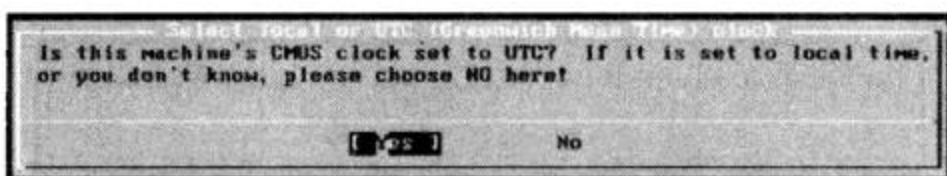


图 2-33 选择系统的 CMOS 时区为 UTC 制式

(31) 选择系统时区，这里的时区跟 Centos5.5 中一样，都是“Asia/ShangHai”，选择顺序依次为“Asia”→“China”→“east China-Beijing, Guangdong, Shanghai, etc.”。最后要选择开启 CST 时区。安装 FreeBSD8.1 的时候，没有配置时区，系统默认的是 UTC 时区，与中国时区相差 8 小时，所以这里要选择将 UTC 时区转成 CST 时区，如图 2-34 所示。

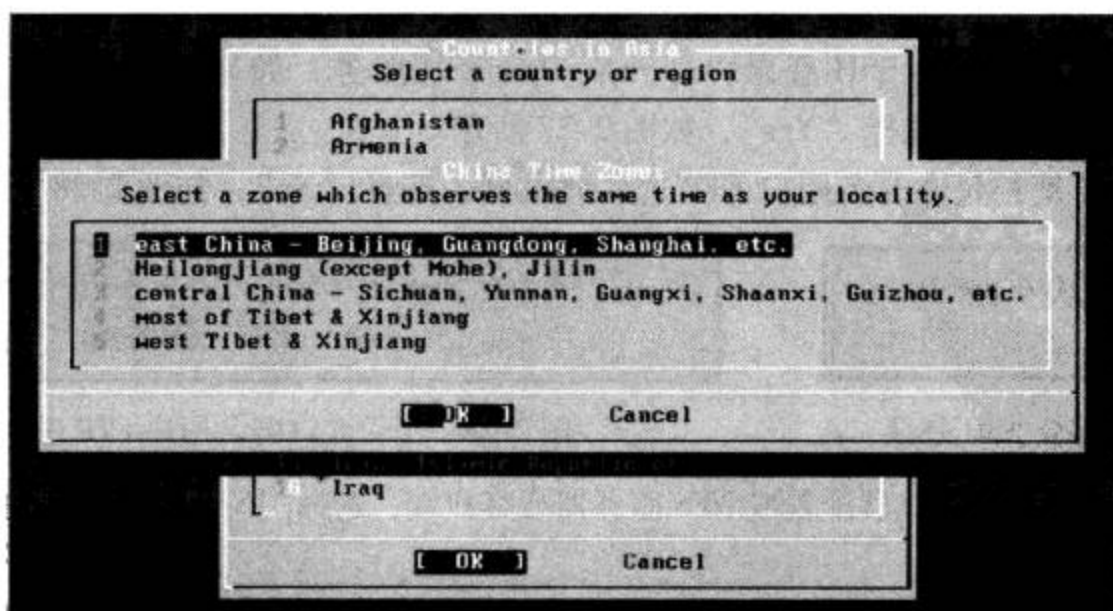


图 2-34 配置系统时区界面

(32) 系统询问是否为此机器配置鼠标，因为是纯字符操作，这里选择“No”，如图 2-35 所示。

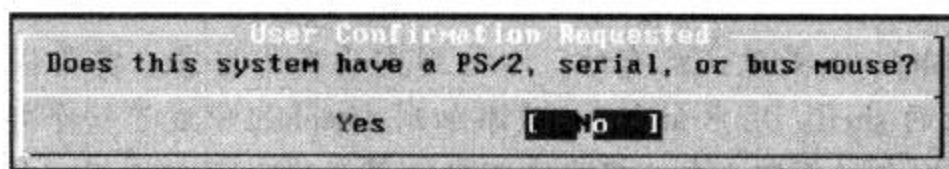


图 2-35 是否需要配置鼠标选项

(33) 进行到这里，如果需要，还可以在这一阶段加入其他的 package。不过安装完成之后，sysinstall 依然可以用来安装其他 package。这里不需要加入，直接选择“No”，如图 2-36 所示。

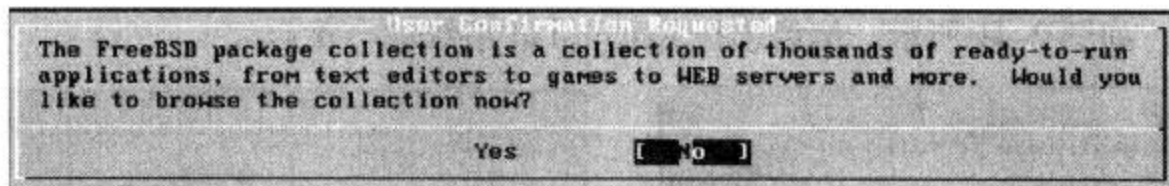


图 2-36 是否继续安装其他软件包

(34) 在安装系统的过程中，系统会提示我们至少添加一个用户，以避免直接以 root 用户的身份登录，如图 2-37 所示。

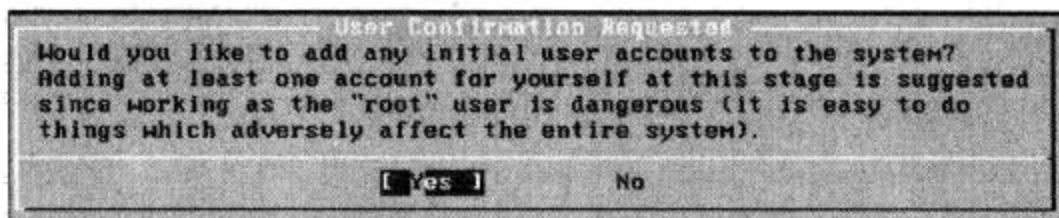


图 2-37 是否允许添加用户

(35) 下面我们就要添加一个普通用户了，记得将此用户添加进“wheel”组（root 系统组）里，这样做的好处是我们可以添加一个具有系统组权限的用户来对系统进行操作。由于 FreeBSD 系统默认是不允许 root 进行远程登录的，而我们用此用户服务可以远程 SSH 登录进行管理服务器的工作，如图 2-38 和图 2-39 所示。

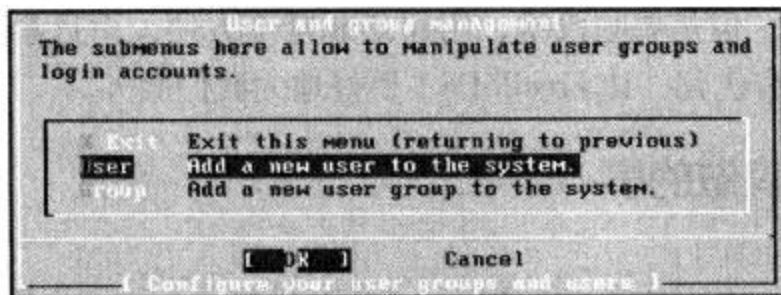


图 2-38 选择添加普通用户选项

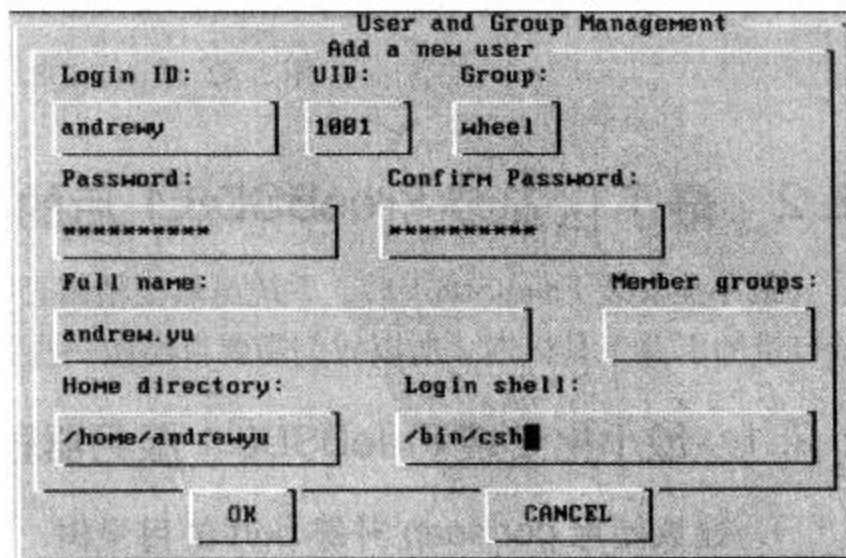


图 2-39 添加普通用户 andrewy 的配置界面

注意 为了安全起见，FreeBSD8.1 默认是不允许以 root 用户登录的，我们可以先用 andrewy 登录系统，然后再切换到 root 下。为方便起见，我们可以将用户登录的 shell（默认是/bin/sh）改为/bin/csh 或/usr/local/bin/bash，csh 是默认就有的，bash 则需要我们安装。需要提醒的是，不要使用一个不存在的或不能登录的 shell。用户 andrewy 被添加到 wheel 组中成了一个超级用户，从而拥有了 root 权限，这样做是因为我们后期要在此用户名下做大量的工作，但又不想切换到 root 下操作，有了 sudo 权限，就可以直接用 sudo 来工作了。

(36) 接下来就是配置 root 密码了，如图 2-40 所示。

(37) 系统最后又会问一次是否还需要全部看一下或更改配置，这是最后一个更改配置的机会了，选择“No”，如图 2-41 所示。然后选择 Exit Install 系统就会重启。

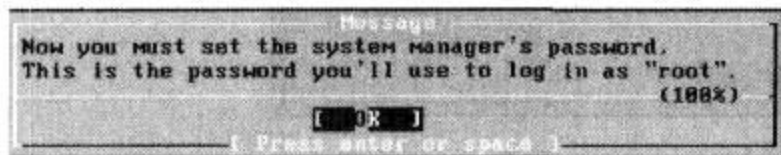


图 2-40 设置 root 密码

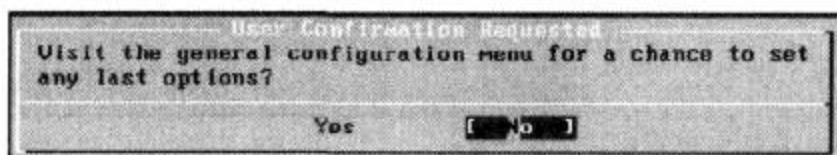


图 2-41 系统咨询是否查看配置

(38) 顺利进入了 FreeBSD8.1 的系统界面，这表示 FreeBSD8.1 已成功安装了，如图 2-42 所示。



图 2-42 FreeBSD8.1 系统安装成功界面

2.2 最小化安装 FreeBSD8.1 后的升级优化部署

最小化安装 FreeBSD8.1 后，系统虽然很精简且稳定，但是要啥没啥，用起来非常不方便，而且系统自带的下载工具极慢，所以我们需要自己做一些更改和优化，让 FreeBSD8.1 更好地为我们服务。

2.2.1 最小化安装 FreeBSD8.1 服务器后建议做的事

1. 建议使用 portsnap 升级 port 的目录树

我们没有使用 portsnap 升级 port 的目录树，这导致我们在直接使用 port 安装 vim 系统时报错，安装平时所用的最基本的的应用软件时同样也会报错。看来真不能偷懒，升级系统后的第一步就应该

做这件事。

(1) 要使用 portsnap, 首先要设置一下它的配置文件, 位于/etc/portsnap.conf 中, 如下所示:

```
[root@bsd01 /usr/ports]# vim /etc/portsnap.conf
SERVERNAME = portsnap.freebsd.org
```

将其修改如下:

```
SERVERNAME = portsnap.hshh.org
```

可以根据实际速度来判断是否需要更新 port 的升级源, 我建议大家还是更新比较好, 新源的速度明显较快, 采用新源不会在下载时出现停顿的现象。

(2) 首次使用 freebsd 的 portsnap 时必须执行下面两步:

```
portsnap fetch
portsnap extract
```

这两步也可以合成为:

```
portsnap fetch extract
```

portsnap fetch 是从网上获取 portsnap 快照的最新压缩包, 听说这个压缩包官方每小时更新一次。portsnap extract 则是把这个压缩包创立到/usr/ports 中, 即使你以前已经手动安装了 ports, 它也会重新创立一次。

(3) 以后更新时我们可以使用 portsnap 更新 ports, 并执行如下两个步骤:

```
portsnap fetch
portsnap update
```

这两步也可以合成使用, 如下所示:

```
portsnap fetch update
```

portsnap 第一次运行 extract 命令时可能需要一段时间, 以后更新时速度就快多了。

2. 配置 make.conf 文件

FreeBSD8.1 中的 ports 默认安装工具是 fetch, 在下载时经常出现速度极慢的现象。为了加快 ports 的安装速度, 推荐使用 axel 工具, 当然具体还须配置 make.conf 文件。

```
cd /usr/ports/ftp/axel
make install clean
```

修改/etc/make.conf, 我习惯用 Vim 来做此项工作, 如下所示:

```
vim /etc/make.conf
```

加入以下内容:

```
FETCH_CMD = axel
FETCH_BEFORE_ARGS = -n 10 -a
FETCH_AFTER_ARGS =
DISABLE_SIZE = yes
MASTER_SITE_OVERRIDE? = \
http://ports.hshh.org/${DIST_SUBDIR}/\
```

```
http://ports.cn.freebsd.org/ ${DIST_SUBDIR}/\
ftp://ftp.freebsdchina.org/pub/FreeBSD/ports/distfiles/ ${DIST_SUBDIR}/
MASTER_SITE_OVERRIDE? = ${MASTER_SITE_BACKUP}
```

以上路径是为了用速度比较快的网站代替程序默认的下源，以达到加速的目的。我的 Server 也安装了 FreeBSD8.1 x86_64，在没有配置前大约是 81KB/s，配置之后是 800KB/s，速度相差悬殊，所以我强烈建议大家配置。

3. 给普通用户增加 sudo 权限

普通用户使用 sudo 的好处是：

- 默认屏蔽了 root，有必要的我们还可以将 root 改名。
- 安装时会询问的问题较少。
- 用户不需要记住额外的密码，他们很可能会忘记 root 的密码。
- 交互式的默认登录避免了“我可以做任何事情”的情况发生，在修改变化前将被提示输入一个口令，这样可以让你考虑一下这么做的结果。

以下安装操作直接在服务器上以 root 身份完成：

```
pkg_add -r -v sudo
```

或者采取 ports 的安装方式，如下所示：

```
cd /usr/ports/security/sudo &&make install clean
```

安装完后记得为此用户赋予 root 一样的权限，语法与 root 的权限分配一样：

```
vim /usr/local/etc/sudoers
root ALL = (ALL) ALL
andrewy ALL = (ALL) ALL
```

这项工作建议在将机器送进机房前完成，不然，如果你在操作时不小心将 FreeBSD 的重要文件 /usr/local/etc/sudoers 破坏了，那么你会发现 FreeBSD 成了一个不受权限控制的机器，犹如断了线的风筝一样在网络的世界里飘移，所以此项操作一定要慎重。

4. 安装 vim7.3

工欲善其事，必先利其器。我还是习惯使用 vim，相信大家跟我一样，所以我们先安装 vim。我是采用 ports 安装的，安装方法如下：

```
cd /usr/ports/editors/vim
sudo make install clean
```

喜欢用 pkg_add 命令的朋友也可以用如下命令来安装：

```
pkg_add -r -v vim-lite
```

安装完成后，输入 vim 就会显示 vim 的功能界面，如图 2-43 所示。

这里补充说明一个细节问题，之所以选择相对而言最新的 vim7.3，是因为 vim7.2 有一个 bug，

```
VIM - Vi IMproved
      version 7.3.121
    by Bram Moolenaar et al.
Vim is open source and freely distributable

  Become a registered Vim user!
type  :help register      for information
type  :q                  to exit
type  :help               or   for on-line help
type  :help version7      for version info

  Running in Vi compatible mode
type  :set nosp           for Vim defaults
type  :help cp-default    for info on this
```

图 2-43 vim 的功能界面

它与 screen 结合起来使用时，打开一个文件就会有假死现象，必须最小化窗口然后再恢复正常窗口才能正常使用。而 vim7.3 修复了这个 bug，所以我这里建议安装 vim7.3。

5. 用 vim 的模板文件优化 vim

编辑用户 andrewy 的 vim 模板文件来优化 vim，可便于以后的编辑工作。如果需要修改对应用户的 vim，则可以到其对应的用户目录中修改 vim 的配置。vim 的配置文件大家都应该有所了解，它存在于用户的 home 目录中，比如说我们要修改 andrewy 用户的 vim，修改 /home/andrewy/.vimrc 文件即可；如果我们要修改 root 的 vim，则可以修改 /root/.vimrc 文件。这里以 andrewy 用户来举例说明，如下所示：

```
vim /home/andrewy/.vimrc
set nobackup
set number
set cindent
set autoindent
set shiftwidth=2
set tabstop=2
set softtabstop=2
set expandtab
set ruler
syntax on
```

下面详细讲解一下这些常用的设定及其具体含义：

- ☐ set nobackup：不要备份文件。使用 backup 备份文件（原文件加后缀 ~）。
- ☐ set number：显示行号。
- ☐ set cindent：设定 C 风格缩进，使用 nocindent 取消设置。
- ☐ set autoindent：设定自动缩进，每行缩进与上一行相等，使用 noautoindent 取消设置。
- ☐ set shiftwidth=2：设定缩进为两个空格。
- ☐ set tabstop=2：设定制表符为两个空格。
- ☐ set softtabstop=2：设定软制表符为两个空格。
- ☐ set expandtab：缩进和（软）制表符使用空格替代，用 noexpandtab 取消设置。
- ☐ set ruler：显示光标所在行列号。
- ☐ syntax on：启动语法高亮。

上面这些设定中涉及的名词术语及相关解释如下所示。

- ☐ cindent：使用 C 语言的缩进方式，根据特殊字符（如 “{”、“}”、“:”）和语句是否结束等信息自动调整缩进，在编辑 C/C++ 等类型文件时会自动设定。
- ☐ softtabstop：软制表符宽度，设置为非零数值后使用 [Tab] 键和 [Backspace] 键时光标移动的格数等于该数值，但实际插入的字符仍受 tabstop 和 expandtab 控制。

6. 更改当前用户的 shell 为 bash

FreeBSD8.1 下默认的 shell 为 sh，使用起来非常不方便，我还是比较喜欢用 bash。要使用 bash，需要先安装，这里用 ports 来安装，如下所示：

```
cd /usr/ports/shells/bash
```

```
sudo make install clean
[andrewy@ jail shells] $sudo chsh -s /usr/local/bin/bash
chsh: user information updated
```

如果显示以上信息，请注销并重新登录。

```
[root@ FreeJail ~]# echo $SHELL
/usr/local/bin/bash
```

如果显示以上信息，则表示 shell 已经成功修改了。另外，建议优化一下用户的 bash，让其可以更方便地工作，如下所示：

```
[andrewy@ jail shells] $sudo vim ~/.profile
```

在最后一行添加如下内容：

```
PS1 = "\[\e[37;40m\][\[\e[32;40m\]\u\[\e[37;40m\]@ \h \[\e[35;40m\]\W\[\e[0m\]]\n $\[\e[33;40m\]"
```

这行代码是给终端添加色彩的，避免过于单调，效果非常好，如图 2-44 所示。



图 2-44 优化后的 SHELL 显示界面

7. 更改默认 csh 的配置

更改默认 csh 的配置，可让我们更方便地在 FreeBSD8.1 下操作。如果有的朋友喜欢用 FreeBSD8.1 下自带的 csh，也可以适当地优化，以方便工作。

如何让 FreeBSD8.1 的 csh 也像 bash 那样按 [Tab] 键就可列出无法补齐的候选文件呢？标准的方法是按快捷键 [Ctrl + D]。下面以 andrewy 用户为例来说明如何用 [Tab] 键列出无法补齐的候选文件。我们可以通过编辑用户 andrewy 的 csh 控制文件 .cshrc 来达到用 [Tab] 键列出无法补齐的候选文件的目的，如下所示：

```
vim /home/andrewy/.cshrc
set autolist
```

如何让此更改立即生效呢？我们可以用 source .cshrc 来实现，如下所示：


```
source .cshrc
```

如果 root 用户也是 csh 环境，则可以在 /root/.cshrc 中执行同样的操作，如下所示：

```
sudo su -
```

切换到 root 用户，修改 .cshrc 文件，增加如下代码：

```
set autolist
```

让其立即生效，如下所示：

```
source .cshrc
```

8. 新安装的命令无法运行怎么办

FreeBSD8.1 的 cshell 会缓存环境变量 PATH 中指定目录中的可执行文件，以加快查找速度，但这会导致一些新安装的命令无法运行，在执行以下命令后才能运行新安装的命令（最典型的例子就是刚安装的 vim 居然提示找不到命令）。

```
rehash
```

通过以上配置我们会发现，FreeBSD8.1 现在用起来越来越方便了，通过 PicTTY + Screen 就可以很轻松地在 FreeBSD8.1 中完成我们的日常工作了。

2.2.2 系统管理员应该知道的 FreeBSD8.1 的一些事项

(1) FreeBSD 可以直接用命令升级（比如将 FreeBSD8.0 升级到 FreeBSD8.1），这是其他服务器系统无法比拟的。下面在测试环境下，以一台 32bit FreeBSD8.0 做测试，我们可以直接在 FreeBSD 的命令行模式下输入以下命令：

```
sudo freebsd - update - r 8.1 - RELEASE upgrade
```

中间按提示输入就行了，如下所示：

```
Does this look reasonable (y/n)?
```

我们全输入“y”，即肯定以上操作。大部分都不需要修改，只是文件的版本时间要改变一下。另外，还有一些需要合并的文件，程序会自动用 vi 编辑器打开，解决一下就行了。

```
sudo freebsd - update install
```

结束后会有如下提示：

```
Installing updates...
```

```
Kernel updates have been installed. Please reboot and run
```

```
"/usr/sbin/freebsd - update install" again to finish installing updates
```

然后执行如下命令：

```
sudo shutdown - r now
```

重新启动后，执行如下命令：

```
sudo freebsd - update install
```

使用 `uname -a` 查看，已经 OK 了。

```
FreeBSD www.wsck.com 8.1 - RELEASE FreeBSD 8.1 - RELEASE #0: Mon Jul 19 02:55:53 UTC 2010 root@
almeida.cse.buffalo.edu:/usr/obj/usr/src/sys/GENERIC i386
```

注意 线上的生产服务器全部用的是 64bit FreeBSD8.0_release 版本，升级过程可能不一样。建议大家先用虚拟机或一些不是特别重要的线上服务器尝试一下，不建议直接用线上服务器升级，请谨慎操作。如果确实有升级的需求，请备份重要资料后再执行此操作。

(2) FreeBSD8.1 的 vim 确实与 Linux 下的 vim 有区别。

用过 FreeBSD 的人都会抱怨 FreeBSD8.1 下的 vim 不好用，特别是与 RHEL5.5 和 Centos5.5 相比较而言，这是不争的事实。不过，如果稍微配置一下 vim，用起来就不会比 Linux 下的 vim 差，具体配置方法请大家参考 2.2.1 节的内容，这里就不重复了。

(3) 在 FreeBSD8.1 下配置网络环境，其实比 Centos5.5 更简单。

使用 `sysinstall` 进行网络配置的优点是所有的网络数据将在同一个界面下进行设置，不容易出现错误和遗漏。熟练的 Unix 用户在平时维护系统时可能更喜欢进行手动配置，因为手动配置有如下优点：

- 熟悉命令之后，手动配置更快。
- 能够使用配置命令的高级特性。
- 更容易维护配置文件，找出系统故障。
- 能更深刻地了解系统配置是如何进行的。
- 如果仅仅使用 `sysinstall` 进行设置，不太可能对系统设置有深刻的了解，一旦遇到问题，很容易束手无策。对于管理员来讲，不能被动地停留在仅能使用和操作的阶段，应该了解系统是如何工作的，这样才能更好地进行系统的维护和管理。因此，需要使用命令行方式进行设置。一旦熟悉之后就会发现，只有命令行才能提供全面而灵活的操作能力，而全屏幕方式则限制很多，过于呆板。另外，为了让配置永久生效，我推荐像 Centos5.5 一样，采用文件配置的方式配置，特别是相对 Centos5.5 下的网卡配置 `/etc/sysconfig/network-scripts/ifcfg-eth0` 而言，感觉更为简便，过程如下（这里以 FreeBSD8.1 为例）：

```
vim /etc/rc.conf
hostname="mail.balaninfo.com"
ifconfig_le0="inet 192.168.1.108 netmask 255.255.255.0"
defaultrouter="192.168.1.1"
```

域名解析 DNS 配置如下：

```
vim /etc/resolv.conf
nameserver 210.5.4.116
nameserver 210.51.176.71
```

为了让以上过程永久生效，reboot 重启服务器后可用 `ifconfig` 验证 IP 地址，`netstat -r` 可验证网关是否生效，`nslookup` 及 `dig` 可验证 DNS 配置是否正确。在这里解释一下名字解析 `hosts`，它的执行顺序优于 DNS，现阶段多用于集群环境，如 Heartbeat 等，配置过程如下：

```
vim /etc/hosts
```

```
192.168.1.100 HA1
192.168.1.101 HA2
192.168.1.188 vip.balaninfo.com
```

事实上，我们现在也将 hosts 应用于没有 DNS 域名服务器的内网开发环境下，也非常方便。

(4) FreeBSD8.1 可以通过更改配置的方法让 root 远程登录，我们是通过修改 SSH 的配置文件/etc/ssh/sshd_config 来实现的，如下所示：

```
vim /etc/ssh/sshd_config
```

在最后添加如下内容：

```
PermitRootLogin yes          #允许 root 登录
PermitEmptyPasswords no      #不允许空密码登录
PasswordAuthentication yes    #设置是否使用口令验证
```

重新启动 ssh 服务，如下所示：

```
/etc/rc.d/sshd restart
```

(5) 强烈建议在 FreeBSD8.1 下采取 sudo 操作，可能有些用 Linux 的读者不太习惯，其实你用了就会发现，是利大于弊的。

虽然在 FreeBSD 下可以配置成允许 root 远程 SSH，但强烈建议不要这么做，最好还是分配一个有 sudo 权限的普通用户进行操作。如果有特殊需求，就必须采取 sudo 操作，这样不仅安全，而且在遇到毁灭性故障时，起码可以起到一点预警的作用。所以我强烈推荐在 FreeBSD8.1 下执行 sudo 操作。

sudo 是一款开源的安全工具，它允许管理员给予某些用户以 root 用户或其他用户身份运行特定命令的权利。它还能记录下特定系统用户的命令和参数。sudo 的开发声明，它的基本出发点就是“让人们以尽可能少的权限完成他们的工作”。

(6) PW 是 FreeBSD8.1 下用来创建、删除、修改、显示用户和组的命令行工具，它还有系统用户和组文件编辑器的功能，使用起来非常方便。我们可以用 pw --help 来了解 pw 的帮助文档，如下所示：

```
pw --help
```

PW 的语法如下：

```
pw [user |group |lock |unlock] [add |del |mod |show |next] [help |switches/values]
```

1) 添加用户及管理，其具体语法规则如下：

```
pw [-V etcdire] useradd [name |uid] [-C config] [-q] [-n name] [-u uid] [-c comment]
[-d dir] [-e date] [-p date] [-g group] [-G grouplist] [-m] [-k dir]
[-w method] [-s shell] [-o] [-L class] [-h fd | -H fd] [-N] [-P] [-Y]
[-V etcdire]
```

默认在新增组和用户时使用/etc/pw.conf 作为 pw 配置文件，也可以指到别的文件上。pw.conf 这个文件可以不存在。当使用这个参数时，需要使用-C config，即：[-C config]。

下面详细说明一下 pw 的参数。

□ [-q]：在交互式环境中，这个选项使 pw 支持输出错误信息。

- [-N]: 在添加和更改操作时输出结果, 而不会真正改变组和用户的信息。
- [-Y]: 自动更新 yp 数据库。
- [-n name]: 指定用户名。
- [-u uid]: 指定用户 ID。
- [-c comment]: 用户全称等注释信息。
- [-d dir]: 指定用户 home 目录。
- [-e date]: 账号到期时, 时间格式可以是绝对的“日-月-年”, 年可以为 4 位的数字年, 月可以是数字或英文月份的简写 (Jan、Feb 等); 也可以是相对时间, +n [分时天周月年], n 可以是十进制、八进制 (0 开头) 或十六进制 (0x 开头)。
- [-p date]: 密码到期时间, 时间格式同上。
- [-g group]: 指定组 (组名或组 ID)。
- [-G grouplist]: 指定组列表, 组以空格分开, 如: -G wheel mysql teczm 即把某账号置于这 3 个组内。
- [-L class]: 在用户创建时指定登录等级。
- [-m]: 自动创建用户 home 目录。
- [-s shell]: 指定用户 shell。
- [-w method]: method 包括以下内容。
 - no: 新创建的账号禁止登录。
 - yes: 强制新建账号的密码与账号相同, 这样就不需要输入密码了。这个选项在大批量增加用户账号时有用。
 - none: 强制新建账号使用空密码。
 - random: 生成一个随机密码。

增加参数-h 0 会出现 “password for user username:” 这样的提示, 也就是让你设定 username 的密码。

2) 组操作时的常用选项如下。

- [-M memberlist]: 将用户置于组内, 会替换掉已经存在的用户。
- [-m newmembers]: 添加新用户到组内, 不会替换掉已经存在的用户。

来看一下下面的示例。

- 新建一个用户 bsder 并使用 cshell, home 目录为 [color] /home/bsder [/color], 属于组 [color] wheel [/color], 口令交互输入如下:

```
pw useradd bsder -s /bin/csh -d /home/bsder -m -g wheel -h 0
```

或者也可以用如下命令:

```
pw user add bsder -s /bin/csh -d /home/bsder -m g wheel -h 0
```

注意 前面曾指出增加参数-h 0 会出现 “password for user username:” 这样的提示, 以让你设定 username 的密码。

- 将 bsder 使用的 shell 改为 /bin/sh。


```
pw usermod bsder -s /bin/sh
```

- 将 bsder 置于 test 组内。

```
pw groupmod test -m bsder
```

- 锁定 bsder 用户账号。

```
pw lock bsder
```

- 对 bsder 账号解锁。

```
pw unlock bsder
```

- 显示 bsder 用户属性。

```
pw usershow bsder
```

- 删除 bsder 用户。

```
pw userdel bsder
```

或者是如下命令：

```
pw user del bsder
```

(7) FreeBSD8.1 的远程连接。PieTTY 使用起来比 xshell3.0 更方便，远程 SSH 的 FreeBSD8.1 机器非常快，推荐大家尝试使用。这个工具也是中国雅虎内部员工推荐使用的，我推荐大家采用 PieTTY + Screen 的方式来连接和管理 FreeBSD8.1 机器，而 PieTTY 的操作与 Putty 基本是一样的，由于功能比较简单，这里就不详细说明了。

(8) FreeBSD 下的虚拟机。大家现在较喜欢用 VMware Esxi 来作为 Linux 的虚拟化，但商业软件毕竟是收费的，而且功能越强大，收费越高。其实 FreeBSD8.1 自身就带了 jail 虚拟机，我们已将其用于对 I/O 性能要求不高的生产服务器了（比如网站后台），详细配置将在后面章节重点介绍。

2.2.3 在 FreeBSD8.1 下高效地安装和卸载软件

在 FreeBSD8.1 下我们是用 ports 和 pkg_add 安装软件的，它们的区别在哪里呢？它们有哪些高级用法？以下都是笔者在工作中的总结，特与大家分享一下。

(1) ports 的目录在 /usr/ports 下，可以通过 /etc/make.conf 文件更改。

(2) 用安装 ports 软件时，有可能提示这个包已经安装过了，无法继续安装，我们用提示中的参数来继续安装，如下所示：

```
make install clean FORCE_PKG_REGISTER=yes
```

这样就又能够继续安装了，上面的“yes”不区分大小写。

(3) 用 ports 安装软件时，有时会出现下载的包的 MD5 值不匹配的情况，我们可以用如下命令来强制安装：

```
make install clean NO_CHECKSUM=yes
```

(4) 如何查找并安装一个 ports 软件？如果知道文件的详细名称，可以直接用 whereis 命令。

```
whereis php5
```

如果不知道详细文件名，可以用如下命令：

```
cd /usr/ports
make search name=php
```

假如只知道描述文件名称的关键字，可以用如下命令：

```
cd /usr/ports/
make search key=php
```

(5) 如何查找已安装软件包的信息？可以用以下命令来查看已安装软件的信息：

```
pkg_info | grep php
```

(6) 如何删除一个软件包？`pkg_delete` 软件包的详细名称可以用 `pkg_info grep` 管道命令查出，加上 `-f` 参数，就能够删除别的软件对其有依靠联系的软件包了（`software_name` 为软件包的名字），如下所示：

```
pkg_delete -f software_name
```

(7) 安装了 `ports` 软件后，如何修改配置？有时在安装 `ports` 包时会有蓝色配置选择页，通常选过一次后就不会再显示了。我们可以用如下命令来删除软件原有配置（必须有 `root` 权限）：

```
make rmconfig
```

如果需要重新配置软件，则可以用如下命令（必须有 `root` 权限）：

```
make reconfig
```

(8) 重新安装已安装的 `ports` 软件包，可以用如下命令：

```
make reinstall
```

(9) 删除已安装的 `ports` 软件包，可以用以下的命令：

```
make deinstall
make clean
```

(10) 如果系统默认的下载地址失效了，可以尝试手动下载 `tbz` 软件包，然后用 `pkg_add` 软件包名直接安装。例如，我们想安装 `vim` 这个软件包，可以用如下命令来完成：

```
pkg_add vim
```

`ports` 的原理其实就是源码编译安装，但是在 `FreeBSD8.1` 下更为人性化一些，我们可以选择软件版本来安装。这一点不像 `Centos5.5`，如果要安装更高级的软件，只有到网上去下载源码软件进行编译安装了。而这一切在 `FreeBSD8.1` 下都是智能化的，全自动完成，这一点也是 `FreeBSD8.1` 的优势所在。有许多朋友从 `Free BSD` 系统转到 `Centos` 可能会不太习惯，他们会觉得在 `Centos5.5` 下安装最新的软件非常麻烦，随着对 `FreeBSD` 了解的深入，他们会觉得 `FreeBSD8.1` 下的软件安装方式应该更智能和方便一些。

2.2.4 查看 FreeBSD8.1 的硬件配置

在 `FreeBSD8.1` 下可查看服务器的命令不是很多，这里特地以一台 `Supermicro`（超星）的双四

核、4GB 内存的服务器为例来说明（为了方便演示，直接切换到 root 下进行操作）。

(1) 查看服务器的 CPU 配置，如下所示：

```
dmesg | grep "CPU:"
CPU: Intel(R) Xeon(R) CPU           E5345   @ 2.33GHz (2333.43 - MHz K8 - class CPU)
```

(2) 如果要在 FreeBSD8.1 下查看 CPU 有几个核，有一个很简单的方法，即在输入 top 命令后观察 STATE 后面 C 参数下的数字，如果数字为 7，表示服务器是 8 核，如图 2-45 所示。

```
last pid: 23200; load averages: 0.00, 0.01, 0.00 up 3+21:48:10 08:11:23
25 processes: 1 running, 24 sleeping
CPU: 0.0% user, 0.0% nice, 0.0% system, 0.0% interrupt, 100% idle
Mem: 13M Active, 292M Inact, 350M Wired, 80K Cache, 418M Buf, 3283M Free
Swap: 12G Total, 12G Free
```

PID	USERNAME	THR	PRI	NICE	SIZE	RES	STATE	C	TIME	%CPU	COMMAND
1308	root	1	44	0	12004K	3808K	select	4	0:02	0.00%	sendmail
1382	root	1	44	0	12096K	4088K	select	4	0:02	0.00%	sendmail
1319	root	1	76	0	7952K	1604K	nanslp	0	0:01	0.00%	cron
1393	root	1	76	0	7952K	1608K	nanslp	4	0:01	0.00%	cron
697	root	1	44	0	7024K	1572K	select	0	0:00	0.00%	syslogd
1181	root	1	44	0	6896K	1560K	select	4	0:00	0.00%	syslogd
1386	smmsp	1	44	0	12096K	3908K	pause	0	0:00	0.00%	sendmail
22879	andrew	1	44	0	10220K	2872K	wait	2	0:00	0.00%	bash
1312	smmsp	1	44	0	12004K	3864K	pause	2	0:00	0.00%	sendmail
22878	andrew	1	44	0	10104K	5264K	select	0	0:00	0.00%	sshd
22875	root	1	44	0	10104K	5192K	swait	1	0:00	0.00%	sshd
22945	andrew	1	44	0	10220K	2872K	ttyn	4	0:00	0.00%	bash
1463	root	1	76	0	6892K	1300K	ttyn	6	0:00	0.00%	getty
1464	root	1	76	0	6892K	1300K	ttyn	3	0:00	0.00%	getty
1456	root	1	76	0	6892K	1300K	ttyn	2	0:00	0.00%	getty
1465	root	1	76	0	6892K	1300K	ttyn	7	0:00	0.00%	getty
1461	root	1	76	0	6892K	1300K	ttyn	1	0:00	0.00%	getty

图 2-45 FreeBSD8.1 下 top 的显示界面

(3) 查看服务器的内存，如下所示：

```
dmesg | grep "real memory" | awk '{FS="[()]" } {print $2}'
4096 MB
```

(4) 查看 swap，如下所示：

```
top | grep "swap" | awk '{print $1, $2}'
Swap: 12G
```

(5) 查看硬盘情况，如下所示：

```
diskinfo -vt /dev/ar1s1
/dev/ar1s1
512          # sectorsize
249859298304 # mediasize in bytes (233G)
488006442    # mediasize in sectors
0           # stripesize
32256       # stripeoffset
30376       # Cylinders according to firmware.
255         # Heads according to firmware.
63          # Sectors according to firmware.
            # Disk ident.
```

Seek times:

```
Full stroke:      250 iter in  1.419320 sec =   5.677 msec
Half stroke:      250 iter in  1.386829 sec =   5.547 msec
Quarter stroke:   500 iter in  2.723002 sec =   5.646 msec
Short forward:    400 iter in  2.351737 sec =   5.879 msec
Short backward:   400 iter in  2.095492 sec =   5.239 msec
Seq outer:        2048 iter in  0.175540 sec =   0.086 msec
Seq inner:        2048 iter in  0.171979 sec =   0.084 msec
```

Transfer rates:

```
outside:          102400 kbytes in  1.220464 sec =   83903 kbytes/sec
middle:           102400 kbytes in  1.174074 sec =   87218 kbytes/sec
inside:           102400 kbytes in  1.790082 sec =   57204 kbytes/sec
```

(6) 查看硬盘的分区情况，如下所示：

```
df -h
Filesystem      Size  Used Avail Capacity  Mounted on
/dev/ar1s1a      19G   374M   17G    2%      /
devfs            1.0K   1.0K   0B    100%    /dev
/dev/ar1s1d     189G   12G   162G    7%     /usr
/usr/jails/basejail 189G   12G   162G    7%     /usr/jails/git/basejail
devfs            1.0K   1.0K   0B    100%    /usr/jails/git/dev
fdescfs          1.0K   1.0K   0B    100%    /usr/jails/git/dev/fd
procfs           4.0K   4.0K   0B    100%    /usr/jails/git/proc
```

(7) 查看网络流量，如下所示：

```
sudo systat -i f 1
```

其中，1 表示 1 秒刷新屏幕一次，显示 Traffic 流量平均值、peak 峰值和 total 流量总值。这是一个很实用的命令，显示内容如下：

```
          /0  /1  /2  /3  /4  /5  /6  /7  /8  /9  /10
Load Average

Interface      Traffic              Peak              Total
  lo0  in      0.000 KB/s          0.000 KB/s        215.488 KB
        out      0.000 KB/s          0.000 KB/s        215.432 KB

  em0  in      2.388 KB/s          2.388 KB/s        571.219 MB
        out      0.341 KB/s          0.841 KB/s        760.988 KB
```

(8) 查看内存加载的模块，如下所示：

```
kldstat -v
26 pcib/acpi_pci
    102 cbb/cardbus
    189 mpt_raid
    187 mpt_cam
    190 mpt_user
    311 g_vfs
    186 mpt_core
    326 firmware
```



```

149 g_md
357 g_class
342 wlan
313 g_part
341 wlan_wep
340 wlan_tkip
339 wlan_ccmp
338 wlan_amrr
310 g_disk
309 g_dev
312 g_label
343 wlan_sta
325 rootbus
333 if_firewire
331 ether
407 x86bios

```

第1章里所介绍的 Centos5.5 系统下的 `vmstat`、`iostat` 等监控硬件或负载的命令在 FreeBSD8.1 下仍适用，查看进程类的命令也是一样的，这里就不再重复了。

总体来说，FreeBSD8.1 查看硬件的命令不多，也没有 Centos5.5 那样人性化，可能每个系统都有自己的特色吧。FreeBSD 系统也许很单一，但它秉承了 BSD 系统的稳定性及安全性优势，用于线上应用环境和开发环境是很让人放心的。

2.3 在 FreeBSD8.1 下部署 jail 虚拟机

我们的开发环境是以 PHP + MySQL 为主的，而开发系统是以 FreeBSD8.0 和 FreeBSD8.1 为主的，有时我们需要在一台性能比较高的机器上部署 8~10 台虚拟机。经过大量的实践，我们最终还是选择了 FreeBSD8.1 下自带的 jail 虚拟机。

2.3.1 FreeBSD8.1 下的 jail 概述

由于系统管理是一项困难而又令人费解的任务，因此人们开发了一系列强大的工具来让管理员的工作变得更加简单。这些改进通常能让系统以更简单的方式进行安装和配置，并且可以毫无问题地持续运转。这其中，许多管理员希望能够正确地为系统进行安全方面的配置，从而防范安全方面的风险。FreeBSD 系统提供了一个用于改善安全问题的工具：jail。jail 是在 FreeBSD 4.X 中由 Poul-Henning Kamp <phk@FreeBSD.org> 引入的，它在 FreeBSD 5.X 中又有了一系列的改进，这使它成为了一个强大而灵活的系统。目前仍然在对其进行持续的开发，以提高可用性、性能和安全性。

jail 以多种方式改进了传统的 `chroot` 环境概念。在传统的 `chroot` 环境中，只限制了进程访问文件系统的哪些部分，其他部分的系统资源（例如系统用户、正在运行的进程，以及网络子系统）是由 `chroot` 进程与宿主系统中的其他进程共享的。jail 扩展了这个模型，它不仅将文件系统的访问虚拟化，还将用户、FreeBSD 的网络子系统以及一些其他系统资源虚拟化。

除了这些之外，jail 也可以拥有自己的用户和自己的 root 用户。当然，这里的 root 用户的权力会受限于 jail 环境，并且，从宿主系统的观点来看，jail 中的 root 用户并不是无所不能。此外，jail

中的 root 用户不能执行除 jail 环境之外的系统中的一些关键操作。更多有关 jail 的细节问题，请参考 FreeBSD 的使用手册，地址如下：http://cnsnap.cn.freebsd.org/doc/zh_CN.GB2312/books/handbook/#。

2.3.2 FreeBSD8.1 下安装 jail 的详细步骤

通过长期的使用和观察，我发现 FreeBSD 8.1 自带的 jail 确实在稳定性和开发上没有什么问题，但性能上与商业虚拟软件相比还是有差距的。编写相关内容时，我有一批安装了 FreeBSD8.0 的机器已将其用于线上环境了，目前运行非常稳定。这里以 FreeBSD8.1 x86_64 为例，详细讲解一下在 FreeBSD8.1 下安装和部署 jail 的步骤（由于 jail 操作要求的权限很高，为了方便，这里直接以 root 操作）。安装前的准备工作如下：

- 宿主机的性能尽量高些，内存越大越好。
- /usr 目录越大越好，我分配的/usr 大约为 300 ~ 400GB。
- 为了达到权限要求和安装的便利，我的操作均是以 root 进行的。

jail 的 IP 跟我的宿主 IP 分别为 192.168.43.128 和 192.168.43.129，这里暂时以一台 jail 的安装为例说明。安装过程如下：

(1) 为 jail 选择路径。Jail 安装的路径是在宿主系统中 jail 的物理位置。一种常用的选择是 /usr/jail/jailname，此处 jailname 是 jail 的主机名。对于“完整”的 jail 而言，它通常包含了 FreeBSD 默认安装的基本系统中每个文件的副本。这里创建 Apache 的目的是做一个 Apache 服务的 jail，如下所示：

```
mkdir -p /usr/jail/apache
```

(2) 编译源码，我们要先进入 /usr/src 目录里，如下：

```
cd /usr/src
```

/usr/src 可以依次选择 sudo sysinstall→Configure→Distributions→src→ALL→Install from a FreeBSD CD/DVD，这样速度快。然后我们可以用 make buildworld 来编译一个新的内核，如下所示：

```
make buildworld
```

make buildworld 的作用是用来建立一个新内核。为什么呢？因为系统安装的内核是一个 generic 内核，一个普遍适用，包含了很多对于本机无用或永远都用不着甚至影响本机性能的东西。所以，为了让它能够发挥最佳的性能，我们采用 make buildworld 建立一个适合自己的内核。

(3) 新建 jail 的源码 world，如下所示：

```
make installworld DESTDIR=/usr/jail/apache
```

(4) 安装配置文件，如下所示：

```
make distribution DESTDIR=/usr/jail/apache
```

distribution 这个 make target 将安装全部的配置文件，换句话说，就是将 /usr/src/etc/ 复制到 jail 环境中的 /etc 上。安装 jail 的过程也是熟悉 FreeBSD 目录结构的过程。

(5) 安装 devfs。在 FreeBSD 系统中，几乎所有的应用程序都至少需要访问一个设备，这主要取决于应用程序的性质和目的。控制 jail 中能够访问的设备非常重要，不正确的配置很可能给

攻击者可乘之机。通过 devfs 实施的控制，可以通过由联机手册 devfs 和 devfs.conf 介绍的规则集配置来实现。但为了以后能方便地 SSH 到 jail 上，这里建议直接在宿主机的 /etc/rc.conf 里添加如下内容：

```
jail_apache_devfs_enable = "YES"
```

这句命令的作用是在 apache_jail 中挂接 devfs。

(6) 配置宿主机的 /etc/rc.conf，我们用 vim /etc/rc.conf 命令，在文件中添加如下内容：

```
jail_enable = "YES"
jail_list = "apache"

jail_apache_rootdir = "/usr/jail/apache"
jail_apache_hostname = "www.cn7788.com"
jail_apache_ip = "192.168.43.129"
jail_apache_exec = "/bin/sh /etc/rc"
jail_apache_devfs_enable = "YES"
```

/etc/rc.conf 里原有的内容如下：

```
ifconfig_le0 = "inet 192.168.43.128 netmask 255.255.255.0"
defaultrouter = "192.168.43.2"
hostname = "mail.cn7788.com"
ifconfig_le0_alias0 = "inet 192.168.43.129 netmask 255.255.255.0"
```

此行的目的是为 Apache 的 jail 添加 IP。

(7) 如何在不重启机器的情况下使此 jail 立即生效呢？可以执行如下命令：

```
sh /etc/rc
```

成功执行以上命令后就可以使用 jls 看到 jail 的状态了。当然 /etc/rc.d/jail 脚本也可以用于手工启动或停止 rc.conf 中配置的 jail，命令如下所示：

```
/etc/rc.d/jail start apache
```

2.3.3 FreeBSD8.1 下 jail 的管理

(1) 管理 jail 时我们可以选择 jexec 工具，先用 jls 找出运行 Apache 的 jid（此例中 apache_jail 的 jid 为 1），然后用 jexec 命令进入此 jail 机器中，如下所示：

```
jexec 1 csh
```

再改变 jail 的 root 密码，如下所示：

```
jexec 1 passwd root
```

(2) 如何启动名称为 Apache 的 jail 的 SSH 呢？可以编辑名称为 Apache 的 jail 下的 /etc/rc.conf 文件，添加如下代码：

```
sshd_enable = "YES"
```

然后重启此 jail 虚拟机，否则不能启动此 jail 机器的 SSH 服务，命令如下：

```
jexec 1 sh /etc/rc
```

另外，在宿主主机上新建一个用于 apache_jail 的 SSH 用户，命令如下：

```
jexec 1 pw useradd admin && jexec 1 passwd admin
```

记得把 admin 放入 wheel 系统组中，这样就可以进去切换 root 账户了。wheel 是系统默认的 root 组，它们有切换到 root 下的权限，命令如下：

```
jexec 1 pw groupmod wheel -m admin
```

(3) 如何让你的 jail 与别的机器 ping 通？在没有配置之前你会发现，无论你 ping 什么，都会出现 “ping: socket: Operation not permitted” 提示。

如果想永久保留配置，可以在宿主主机上面修改 /etc/sysctl.conf 文件，即 vim /etc/sysctl.conf，并在其中添加如下代码：

```
security.jail.allow_raw_sockets=1
```

重启 jail 虚拟机，如下所示：

```
/etc/rc.d/jail restart apache
```

(4) 如何挂载 jail 虚拟机的文件系统呢？应该从宿主主机将 /etc/resolv.conf 文件复制到 jail 系统中，如下所示：

```
cp /etc/resolv.conf /usr/jail/apache/etc/resolv.conf
```

还要将宿主机的 make.conf 也复制过去，这样 ports 的安装速度就很快了，如下所示：

```
cp /etc/make.conf /usr/jail/apache/etc/make.conf
```

在宿主主机上将 /usr/ports 挂接到 jail 上，下面这行代码可添加到 /etc/rc.conf 上：

```
mount_nullfs /usr/ports /usr/jail/cas/usr/ports
```

在使用 jail 的过程中发现，用以上方法新建单台 jail 并没有问题，但如果要新建 10 台以上的 jail 虚拟机，过程不仅麻烦而且很费时间。所以推荐采用 jail 的优化管理工具 ezjail，下面将详细介绍。

2.3.4 通过 ezjail 来创建和管理 jail 虚拟机

在公司的环境中，我需要快速地部署大量 jail 虚拟机，原先的 make world 方式就显得非常力不从心了。因为大家都知道，make world 在双四核的机器上也是极慢的。查阅了 FreeBSD 的官方文档，它向我们推荐管理及创建 jail 的工具 ezjail。

jail 的手册介绍了创建 jail 的方法，然而，当你需要多个 jail 时，完整的 jail 目录树会迅速地占用很多宝贵的磁盘空间。ezjail 通过使用 FreeBSD 的 nullfs 特性来避免这个问题。基础系统的大部分 (/bin、/boot、/sbin、/lib、/libexec、/rescue、/usr/bin、include、lib、libexec、ports、sbin、share、src) 在宿主主机系统中都共存在一个副本，并通过 nullfs 以只读的方式挂载到所有的 jail 中。那些 jail 就非常“苗条”了（每个约 2MB），并且只包含一些指向 basejail 挂载点的软链接，没有像 /etc、/usr/local 这样的共享目录。ezjail 会带来如下优势：

- 节省磁盘空间、inode 和内存，因为系统只需要为所有的 jail 持有一个系统二进制的副本。
- 可以通过更新一个基础目录来更新所有的 jail，它是如此简单，你肯定能做到。
- 入侵者破坏 jail 时无法安装标准的 rootkit（因为基础系统是只读挂载的）。
- 由于 ezjail 完全是用 sh 写成的，因此没有必要在宿主机系统中安装其他脚本语言。
- 因为基础系统是软链接的，被“囚禁”（enjailed）的用户可以选择不使用已挂载的 world。
- 一个常常被低估的事实是：较少的复杂性意味着更多的安全。

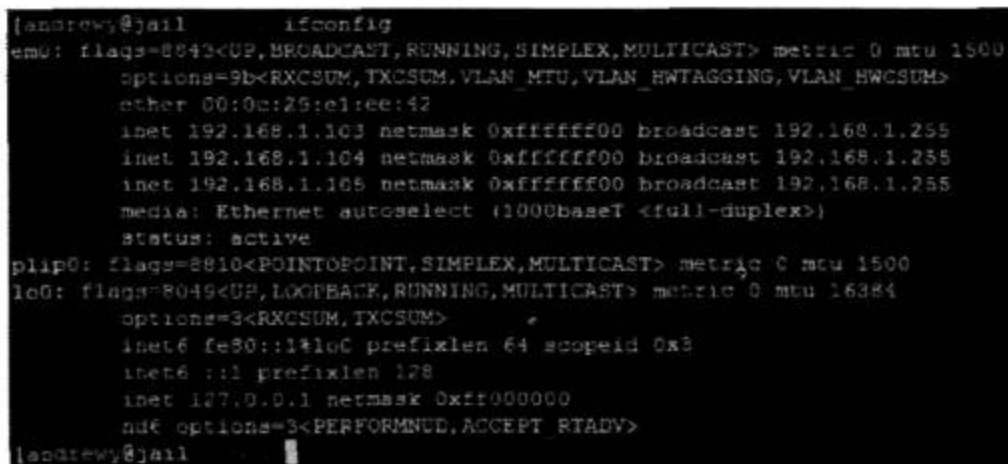
我现在想通过 ezjail 来为一台名为 jail.cn7788.com、IP 为 192.168.1.103 的机器（相当于母机）安装一台名为 apache.cn7788.com、IP 为 192.168.1.104 和一台名为 research.cn7788.com、IP 为 192.168.1.105 的 jail 机器。

具体步骤如下（由于权限要求很高，为了方便，这里暂时切换到 root 下操作）。

(1) 编辑网卡地址，为网卡创建两个子网地址，在/etc/rc.conf 中添加如下代码：

```
ifconfig_em0_alias0="inet 192.168.1.104 netmask 255.255.255.0"
ifconfig_em0_alias1="inet 192.168.1.105 netmask 255.255.255.0"
```

重启后，通过 ifconfig 观察网卡的 IP，如图 2-46 所示。



```
[andrew@jail ~]$ ifconfig
em0: flags=8043<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
    options=9b<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_HWCSUM>
    ether 00:0c:29:e1:ee:42
    inet 192.168.1.103 netmask 0xfffff00 broadcast 192.168.1.255
    inet 192.168.1.104 netmask 0xfffff00 broadcast 192.168.1.255
    inet 192.168.1.105 netmask 0xfffff00 broadcast 192.168.1.255
    media: Ethernet autoselect (1000baseT <full-duplex>)
    status: active
plip0: flags=8010<POINTOPOINT,SIMPLEX,MULTICAST> metric 0 mtu 1500
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> metric 0 mtu 16384
    options=3<RXCSUM,TXCSUM>
    inet6 fe80::1%lo0 prefixlen 64 scopeid 0x3
    inet6 ::1 prefixlen 128
    inet 127.0.0.1 netmask 0xff000000
    nd6 options=3<PERFORMNUD,ACCEPT_RTADV>
[andrew@jail ~]$
```

图 2-46 ifconfig 显示网卡详细信息界面

(2) 编译内核。由于是最小化安装，所以没有/usr/src 目录，我们可以通过 sysinstall 工具来安装此目录。先切换到 root 下，然后输入 sysinstall 命令，再按顺序执行以下步骤以安装此目录：

sysinstall→Configure→Distributions→src→ALL→Install from a FreeBSD CD/DVD

然后是编译内核，命令如下：

```
cd /usr/src
make buildworld
```

此过程比较漫长，建议在下班后或午饭时间做。无论是双四核还是双八核的机器，都要运行 2 小时以上。

(3) 安装 ezjail 工具，如下所示：

```
cd /usr/ports/sysutils/ezjail
make install clean
```

(4) 生成 jail 模板，如下所示：

```
ezjail-admin update -p -i
```

其中 p 表示提供给 jail ports; i 表示不再运行 make world, 因为第一步我们已经做了。

(5) 生成名为 apache.cn7788.com 和 reseach.cn7788.com 的子 jail 机器, 如下所示:

```
ezjail - admin create -r /usr/jails/apache apache.cn7788.com 192.168.1.104
ezjail - admin create -r /usr/jails/research research.cn7788.com 192.168.1.105
```

该命令分别可以在 /usr/jails/apache 和 /usr/jails/research 目录下建立名为 apache.cn7788.com 和 research.cn7788.com 的 jail 机器。

(6) 让 192.168.1.103 的机器开机, 即启动 ezjail 工具, 在 /etc/rc.conf 的最后添加如下代码即可:

```
ezjail_enable = "YES"
```

(7) 分别通过 ezjail 启动这两个 jail 机器, 如下所示:

```
/usr/local/etc/rc.d/ezjail.sh start apache.cn7788.com
/usr/local/etc/rc.d/ezjail.sh start research.cn7788.com
```

这里补充说明一点: ezjail 由两个脚本组成, 即 ezjail-admin 和 ezjail.sh, 前者用于创建、更新和删除 jail, 后者用于启动、停止和重启 jail。

(8) 通过 ezjail-admin list 查看 jail 机器的情况, 如下所示:

```
ezjail - admin list
```

STA	JID	IP	Hostname	Root Directory
DR	2	192.168.1.105	research.cn7788.com	/usr/jails/research
DR	1	192.168.1.104	apache.cn7788.com	/usr/jails/apache

(9) 启动 jail 机器的 SSH。启动 SSH 的方法与平常一样, 没有任何区别。先通过 jexec 1 或 2 sh 登录该 jail, 执行以下命令就可以自动启动 SSH 服务了, 如下所示:

```
echo 'sshd_enable = "YES"' >> /etc/rc.conf
```

然后通过 ezjail.sh 命令重新启动虚拟机, 如下所示:

```
/usr/local/etc/rc.d/ezjail.sh restart apache.cn7788.com
Stopping jails: apache.cn7788.com.
Configuring jails:..
Starting jails: apache.cn7788.com.
usr/local/etc/rc.d/ezjail.sh restart research.cn7788.com
Stopping jails: research.cn7788.com.
Configuring jails:..
Starting jails: research.cn7788.com.
```

(10) 如果要添加新的 jail 机器, 重复以上步骤即可。

(11) jail 机器的备份是非常容易的, 对相应的目录进行备份即可。笔者建议只对重要资料进行备份, 如果 jail 机器崩溃, 重建也非常快。

(12) 如果大家有大规模部署虚拟机的需求的话, 建议用 ezjail 的方式创建和管理, 它创建的速度非常快, 而且子 jail 极小, 如果遇到崩溃的情况, 它恢复起来也极快。现在我基本上在每台 jail 机器上都部署了 ezjail。

2.3.5 jail 在生产环境下的注意事项

我们已经将 jail 用在了线上的生产环境，发现其中有许多需要注意的事项，如下所示：

(1) 线上的服务器用的 jail 机还是蛮多的，一般一台 Dell PowerEdge710 至少要装 10 台左右的子 jail 机器，高峰期间是 1000 多个 PHP 程序在运行，所以 CPU 的负载很大。于是 Nagios 很负责地认为这台机器的负载很大了。所以就狂报警，这其实是一种误报，所以我们要将 jail 机器的 CPU 报警阈值适当调大些。

(2) 子 jail 机器多了以后对磁盘的需求很大，所以刚开始规划时请尽量选择大的磁盘，预留空间给 jail。对于那些对磁盘占用特别大的子 jail 机器，要么分配新磁盘，要么就不用 jail 来做。

(3) 机器多了不太方便管理，所以我建议创建和管理 jail 都用 ezjail，它还是很方便的，且比较有效率，尤其适用于部署大规模的虚拟机集群环境。

(4) 在一次 IP 迁移工作中我们发现，如果 jail 宿主机的网关发生更改，则必须要重新启动宿主机才能生效。

(5) 如果是 MySQL 数据库这种磁盘 I/O 读写频繁的应用服务，我不太建议放在 jail 下。不过，如果是开发 MySQL 数据库，由于读写磁盘 I/O 并不频繁，可以放在 jail 下面。

ezjail 在 jail 机器的创建和管理上确实是很有优势的，推荐大家在工作中用此工具来创建和维护 jail 虚拟机。大家将会发现，这样工作起来会更得心应手，事半功倍。

2.4 在 FreeBSD8.1 下搭建版本控制服务器

2.4.1 版本控制软件的概念

版本控制就是要让你及时地发布你的软件，每个版本能完成应该完成的功能。简单点说，你在开发过程中会不断发现新需求，不断发现 Bug，如果不做控制，你的软件将永远不会发布，或今天一个版本，明天又是一个版本。也可以说，版本控制是软件开发的时间魔法师，它可以将软件轻松地回滚到某一个版本上。正因为它是如此的强大和神奇，所以推荐大家掌握它的用法。在下面的小节里，我会分别向大家介绍在 FreeBSD8.1 下安装 CVS、SVN 及 Git 这 3 种免费开源的版本控制服务器的方法。

2.4.2 在 FreeBSD8.1 下搭建 CVS 服务器

CVS 是开放源代码的配置管理工具，其源代码和安装文件可以免费下载。CVS 是源于 Unix 的版本控制工具，要安装和使用 CVS，最好是对 Unix 系统有所了解。CVS 的服务器管理需要进行各种命令行操作。目前，CVS 的客户端有 winCVS 的图形化界面，服务器端也有 CVSNT 的版本，易用性正在提高。CVS 的功能如下：

它的客户机/服务器存取方法使得开发者可以从任何因特网的接入点中存取最新的代码；它无限制的版本管理检出模式避免了通常因为排他检出模式而引起的人工冲突；它的客户端工具可以在绝大多数的平台上使用。同样，CVS 也不提供对变更流程的自动管理功能。

一般来说，CVS 的权限设置单一，通常只能通过 CVSROOT/passwd 文件、CVSROOT/readers 文件和 CVSROOT/writers 文件，还要设置 CVS REPOS 的物理目录权限来完成权限设置，而且无法完成

复杂的权限控制。但是 CVS 通过 CVS ROOT 目录下的脚本，提供了相应功能扩充的接口，不但可以完成精细的权限控制，还能完成更加个性化的功能。

它的缺点如下：由于 CVS 是开放源码软件，没有生产厂家为其提供技术的支持。如发现问题，通常只能靠自己查找网上的资料来解决。

FreeBSD8.1 下安装 CVS 服务器的具体步骤如下：

(1) 准备工作（由于这里涉及的权限很多，因此还是以 root 账号操作）。将 ports 安装升级等做好。由于在安装 CVS 的过程中由于要用到 Apache 的 htpasswd，所以预先要安装好 apache22，命令如下：

```
cd /usr/ports/www/apache22
make install clean
```

先安装 wget，由于 FreeBSD8.1 下最小化安装时没有预装此工具，所以我们要用 pkg_add 来进行安装。其中 -r 参数是递归的意思，它会安装此软件依赖的其他软件；-v 是显示安装过程。

```
pkg_add -r -v wget
```

下载 cvs 源码包。

```
wget http://sourceforge.net/projects/ccvs/files/VS%20Stable%20Source%20Release/1.11.21/cvs-1.11.21.tar.gz/download?use_mirror=nchc
```

源码编译安装 CVS，大家应该对此有所了解，./configure 和 make 及 make install 是 Linux/Unix 系统下的软件源码安装三部曲，--prefix=/usr/local/cvs 会将我们的 CVS 软件安装在/usr/local/cvs 目录下，如下所示：

```
./configure --prefix=/usr/local/cvs
make
make install
```

(2) 设置 CVS 使用超级服务器进程 inetd 启动方式。修改 inetd 的配置文件/etc/inetd.conf，在最后一行添加如下命令：

```
cvspserver stream tcp    nowait root    /usr/bin/cvs    cvs --allow-root=/home/cvsroot pserver
```

注意 这里 root 是执行 cvs 进程的用户，/usr/bin/cvs 是 CVS 可执行文件的存储路径，要保证/usr/bin/里有 CVS 可执行文件。

cvs --allow-root=/home/cvsroot 指定 CVS 的项目主目录，即用户可访问的项目原文件的一个存放目录，它可以指定多个源代码仓库。检查/etc/services 里是否有 cvspserver 选项，若无的话则需要用一个标识来代替 CVS 的端口号，即在/etc/services 文件中添加如下内容：

```
cvspserver 2401
```

表示用 cvspserver 来标识 2401 端口。

(3) 添加 CVS 的管理用户和组。在 FreeBSD8.1 下我们用 adduser 命令，按照提示一步步增加 cvsroot 用户，freebsd 默认为此用户创建一个与用户名同名的组 cvsroot，其他均可设置成默认值。

(4) 初始化 CVS 源代码仓库。将/home/cvsroot 目录的权限改为 775，这在第三步已配置，这里检查一下即可，使在 cvsroot 组的用户对此目录也有读写权限。初始化目录如下：

```
cvs -d /home/cvsroot init
chmod -R 644 /home/cvs/cvsroot/CVSR00T/config
```

给予 config 文件相应的权限，不然等会儿又会因权限问题报错。

必须说明的是，在 CVS 中默认一个用户 checkout 代码的时候，会在当前模块下生成一个锁文件，如果这个用户对当前模块没有写权限，那么读也是不可能的。配合上面的权限设置，必须改一下 CVS 服务器的配置，改成不在当前模块目录下生成锁文件，把锁文件集中到一个所有用户都有读写权限的目录中。修改配置文件 CVSR00T/config，如下所示：

```
# Put CVS lock files in this directory rather than directly in the repository
#LockDir = /var/lock/cvs
```

去掉 LockDir 前的#即可。建立锁文件集中目录，如下所示：

```
mkdir -p /var/lock/cvs
```

给予最高权限，如下所示：

```
chmod -R 777 /var/lock/cvs
```

(5) 为 CVS 添加一个普通用户 test。

adduser 命令添加 test 用户，密码自己设定。用 pw groupmod 将其添加至 cvsroot 组中。

```
pw groupmod cvsroot -m test
```

配置完成后我们可用命令 id test 来检查以上配置是否生效。

创建 CVS 用户密码文件，创建用户 admin，密码为 admin，因其加密方式与 apache 的 htpasswd 相同，c 参数用一次即可，添加下一个用户时就不需要此参数了，如下所示：

```
htpasswd -cb /home/cvsroot/CVSR00T/passwd admin admin
```

继续为 CVS 添加 test 用户，如下所示：

```
htpasswd -b /home/cvsroot/CVSR00T/passwd test test
```

上述命令的意思是添加 test 用户，密码为 test。

(6) 启动 inetd 超级服务器进程，使其监听 2401 端口，如下所示：

```
/etc/rc.d/inetd start
```

重读配置文件/etc/rc.d/inetd reload。

(7) 用 sockstat -4l 查看系统正在监听的端口，查看 2401 端口是否正处于 LISTENED 状态，如下所示：

```
sockstat -4l
```

在 FreeBSD8.1 下用 sockstat -l 或 netstat -an 来查看系统打开的端口，并使用此端口的进程及用户的信息。还可以用 telnet localhost 2401 来验证端口打开没有。

(8) 本机用户验证是否可以登录 CVS 服务器，可以用如下命令：

```
cvs -d:pserver:test@192.168.21.246:/home/cvsroot/ login
```

(9) 在 Centos5.5 或 FreeBSD 下如何远程连接呢？如果要在 SHELL 下用到 export 命令，请注意你的 SHELL 是否为 sh 或 bash，如果不是请切换到 bash 或 sh 下，然后输入如下命令远程连接 CVS 服务器：

```
export CVSROOT=:pserver:test@192.168.21.246:2401/home/cvsroot
cvs login
Logging in to :pserver:test@192.168.238.128:2401/home/cvs/cvsroot
CVS password: test
```

签出项目到本地，如下所示：

```
cvs checkout project
```

下面将介绍一下 CVS 的命令行操作。

创建一个名为 <module-name> 的目录，将模块内容放在目录中，下面两种方法都可以：

```
cvs checkout <module-name>
cvs co <module-name>
```

从服务器中将文件更新到最新版，下面两种方法都可以：

```
cvs update
cvs up
```

添加新文件到 CVS 服务器，如下所示：

```
cvs add <filename>
```

提交此文件到 CVS 服务器，如下所示：

```
cvs commit -m "My message" <filename>
```

查看文件 <filename> 的提交信息，如下所示：

```
cvs log <filename>
```

查看文件的状态，类似于 Locally Modified，如下所示：

```
cvs status <filename>
```

查看文件现有的标签与分支，如下所示：

```
cvs status -v <filename>
```

查看文件的工作版本与分支中最新版本的不同之处，如下所示：

```
cvs diff <filename>
```

在 Windows XP SP3 下我习惯用 TortoiseCVS 来提交代码，配置也比较简单，具体设置如图 2-47 所示。

Windows XP 下的图形界面操作不是本节重点，这里不详细介绍，有兴趣的朋友可自行尝试。有条件的可以跟公司的开发人员交流其详细用法，但还是推荐大家掌握命令行操作。另外，鉴于现在大多数公司的软件控制版本都是 SVN，所以下面会重点介绍如何在 FreeBSD8.1 下搭建 SVN 服务器。

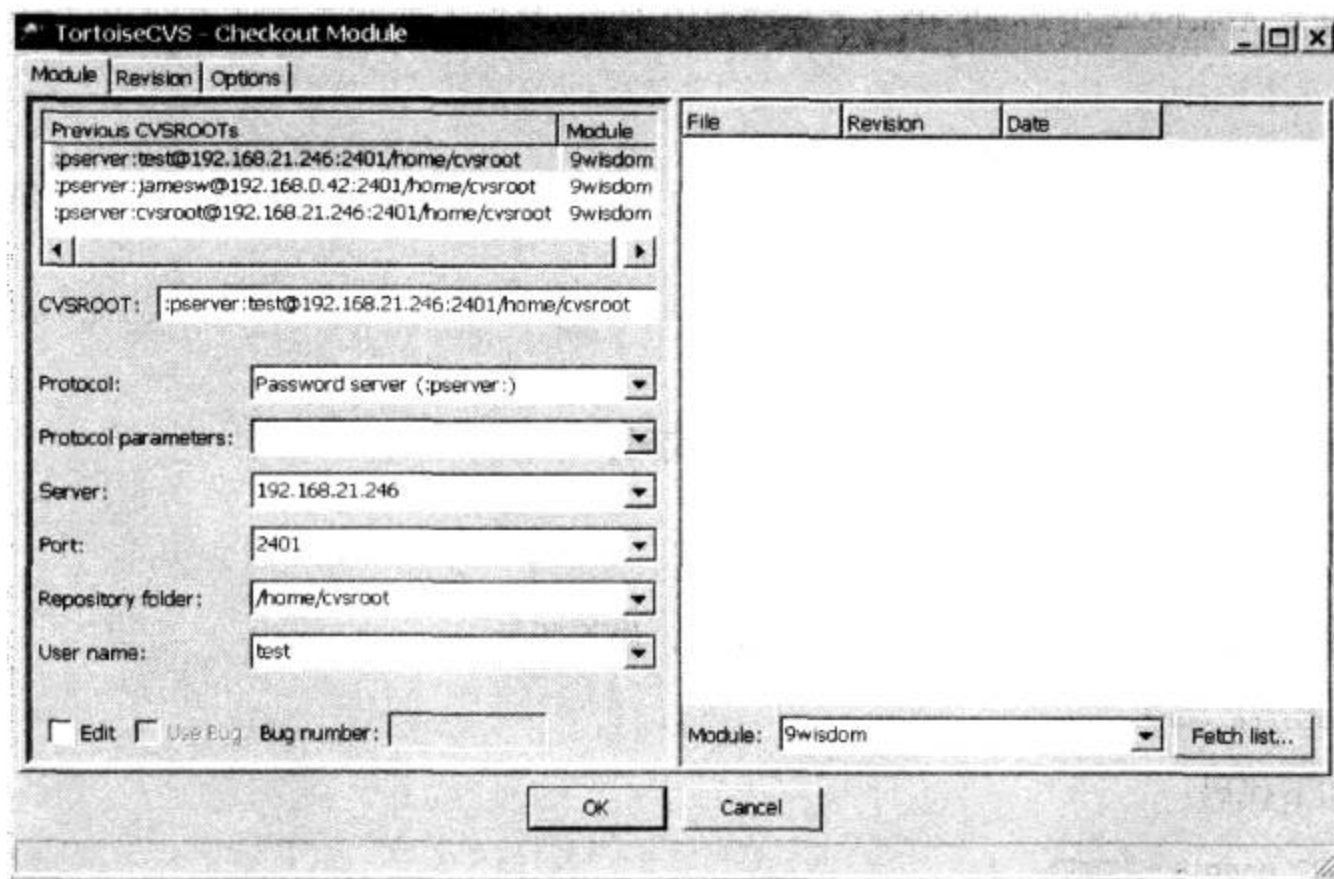


图 2-47 TortoiseCVS 工作界面

2.4.3 在 FreeBSD8.1 下搭建 SVN 服务器

SVN 服务器是版本控制系统（可以看做是 CVS 的升级），一般用于开发环境中，比如多人协同开发项目的源代码管理。由于 SVN 服务器使用方便，所以它现在是版本控制系统的主流，而且它尤其适合集中式的版本管理。另外，Git 是后起之秀，它适用于更大的项目，多用于分布式环境。SVN 版本软件服务器在 FreeBSD8.1 下的配置过程如下所示。

(1) 安装前的准备工作。系统是 FreeBSD 8.1 x86_64，相关优化及配置就不多费笔墨了。将 ports 配置升级完毕，SVN 就可以独立运行了，也可以以 Apache 附加的形式运行。这里为了权限控制的方便，采用的是 Apache 方式，所以这里的第一步是配置 Apache。另外，由于牵涉权限控制的地方比较多，所以这里仍直接以 root 账号操作。

(2) 安装 Apache22，以下为安装过程，命令如下：

```
cd /usr/ports/www/apache22
make WITH_BERKELEYDB=db4 install clean
```

增加的选项就是为了编译安装 subversion 时要用到的 db4 库。subversion 编译安装时有很多的默认配置信息，选择默认信息就可以了。

安装完毕后，我们要将 Apache 配置成自启动模式，如下所示：

```
echo 'apache22_enable="YES"' >> /etc/rc.conf
```

由于习惯了 sed 的流编辑，所以不想用 vim 打开编辑，以上 sed 代码可以简化文件输入过程。

在 FreeBSD8.1 下启动 Apache22 有个小 bug，如果想在 FreeBSD8.0 下安装完 Apache22 就直接启动是不行的，必须要对此 bug 进行修复。

(3) 修复 Apache22 在 FreeBSD8.1 下启动时的 bug, 我们需要安装 accf_data 和 accf_http 软件, 如下所示:

```
cd /usr/src/sys/modules/accf_data
make
make install clean
cd /usr/src/sys/modules/accf_http
make
make install clean
```

然后在/etc/loader.conf 文件的最后加入如下代码:

```
accf_data_load = "YES"
accf_http_load = "YES"
```

添加配置信息并启动, 修改/etc/rc.conf 文件, 如下所示:

```
#vim /etc/rc.conf
```

加入如下代码:

```
apache22_enable = "YES"
apache22_http_accept_enable = "YES"
```

这时候启动 Apache22 就应该没什么问题了, 如下所示:

```
/usr/local/etc/rc.d/apache22 start
```

另外一种解决方法如下:

```
vim /usr/local/etc/apache22/httpd.conf
```

然后用#号注释掉下面这一句:

```
LoadModule unique_id_module libexec/apache22/mod_unique_id.so
```

(4) 安装 SubVersion, 跟前面安装软件的方法一样, 采用 port 安装方法, 如下所示:

```
cd /usr/ports/devel/subversion/
make WITH_MOD_DAV_SVN=yes install clean
```

(5) 配置 Apache22, 让 SVN 以附加方式启动, 如下所示:

```
vim /usr/local/etc/apache22/httpd.conf
<Location /svn>
    DAV svn
    SVNParentPath /data/svn
    AuthzSVNAccessFile /data/svn/svn - acl - conf
    AuthType Basic
    AuthName "Ewizweb SVN Server"
    AuthUserFile /data/svn/svn - auth - file
    Require valid-user
</Location>
```

分配权限给/data/svn 目录, 让其权限完全属于 www: www 这个用户, 如下所示:


```
mkdir -p /data/svn
chown -R www:www /data/svn
chmod -R 755 /data/svn
```

(6) 修改配置用户的权限控制文件 `svn-acl-conf`，它可以控制用户的读写访问权限，如下所示：

```
vim /data/svn/svn-acl-conf
[groups]
bolan_dev=web, andrwy, cc
test=test
[produce:/]
@ bolan_dev = rw
@ test = r
[rest:/]
@ bolan_dev = rw
@ test = r
```

此文件可以控制你的内网中不同的开发小组，只允许他们访问各自的 project。此权限比较简单，R 代表读权限，W 代表写权限。

(7) 配置 SVN 的用户验证文件，如下所示：

```
htpasswd -c /data/svn-auth-file andrwy
```

添加用户，命令如下：

```
htpasswd /data/svn-auth-file test
```

这样，正常启动 Apache22 后，SVN 也正常启动了，这时可以在 Linux、Unix 或者 Windows XP SP3 下通过 SVN 命令或 SVN 客户端连接我们的 SVN 服务器。

1. SVN 的命令行工具

以下命令是开发人员常用的命令，说老实话，我感觉速度比 Windows 下的 TortoiseSVN 快多了，建议大家以命令行为主、TortoiseSVN 为辅来进行 SVN 的版本管理工作。导入项目，命令如下：

```
svn import 代码所在目录 http://192.168.1.102/project --message "Start project"
```

导出项目，命令如下：

```
svn checkout http://192.168.1.102/project 工作目录(自己设定)
```

为失败的事务清场，命令如下：

```
svn cleanup
```

在本地进行代码修改，检查修改状态，命令如下：

```
svn status -v
```

比较不同版本间的差异，命令如下：

```
svn diff
```

更新 (update) 服务器数据到本地，命令如下：

```
svn update directory
```

```
svn update file
```

增加 (add) 本地数据到服务器, 命令如下:

```
svn add file.C
svn add dir
```

对文件进行改名和删除, 命令如下:

```
svn mv b.c bb.C
svn rm d.c
```

提交 (commit) 本地文档到服务器, 命令如下:

```
svn commit
svn ci
svn ci -m "commit"
```

查看日志, 命令如下:

```
svn log directory
svn log file
```

2. Windows XP 下的 TortoiseSVN 图形工具

正常安装 TortoiseSVN 后就可以在 Windows XP 下用 TortoiseSVN 正常连接 SVN 服务器, 并 check-out 及 commit 了。安装完 TortoiseSVN 后它会自动跟你的 IE 集成, 如图 2-48 和图 2-49 所示。

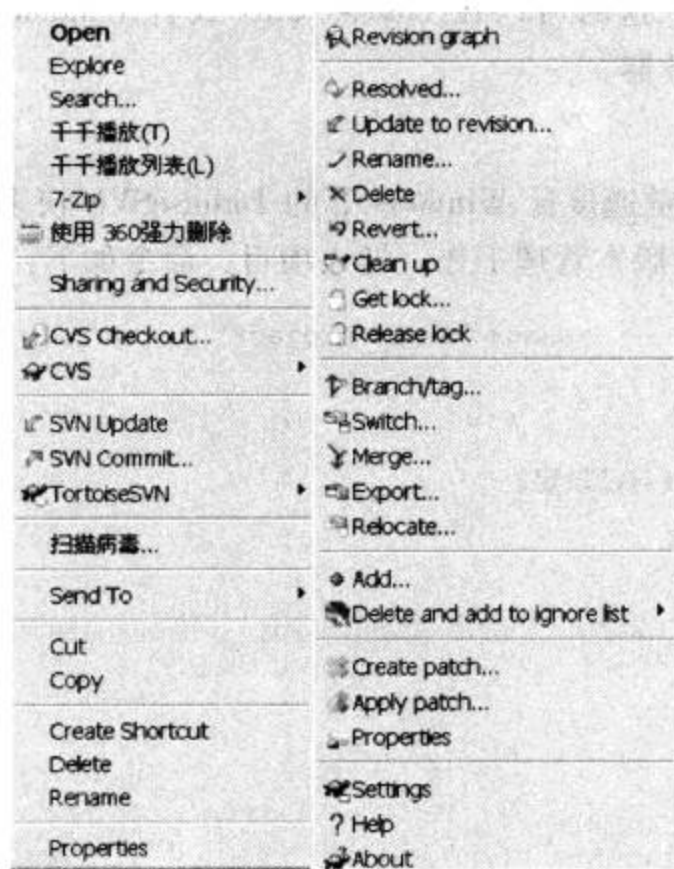


图 2-48 TortoiseSVN 自动与 IE 集成的界面



图 2-49 TortoiseSVN 安装后的 SVN 目录和文件

在 Windows XP 下的图形界面操作不是本节重点, 这里不做详细介绍。还是推荐大家掌握命令行操作, 命令行的速度和效率远优于 TortoiseSVN。

2.4.4 在 FreeBSD8.1 下搭建 Git 服务器

1. Git 简介

Git 是 Linus 为了更好地管理 Linux 内核开发而创立的分布式版本控制/软件配置管理软件。与常用的版本控制工具 CVS、Subversion 等不同，它采用了分布式版本库的方式，不必有服务器端软件的支持，使源代码的发布和交流极其方便。Git 的速度很快，这对于诸如 Linux kernel 这样的大项目来说自然是很重要的，Git 最为出色的是它的合并跟踪（merge tracing）能力。

2. Git 与 Subversion 的对比区分

Subversion 属于集中式的版本控制系统。

- 每个版本库有唯一一个“官方地址”，每个用户都从这个唯一地址获取代码、数据。
- 要获取代码库的更新，也只能连接到这个唯一的代码库，与其同步，以取得最新数据。
- 提交必须有网络连接（非本地版本库）。
- 提交需要授权，如果没有授权，则提交失败。
- 提交并非每次都成功。如果有其他人先于你提交，会提示“改动基于过时的版本，先更新再提交”诸如此类的提示。
- 冲突解决是一个提交速度的竞赛：手快者，先提交，平安无事；手慢者，后提交，可能遇到麻烦的冲突问题。

Git 属于分布式的版本控制系统。

- 众生平等，每个检出（checkout）的版本库，或者更准确地说每个克隆（clone）的版本库都是平等的。
- 你可以从任何一个版本库中克隆创建属于你自己的版本库，同时你的版本库也可以作为源提供他人，只要你愿意。
- 获取版本库的更新，可以来自任何源。
- 你可以从张三那里获得上游的改动，包括张三自己的提交；你也可以从李四那里获得上游的改动，也可能包括李四的提交。
- 提交完全在本地完成，无须别人给你授权，你的版本库你做主。当然，别人是否愿意将你在自己的版本库中所进行的改动合并到他们的版本库中则是另外的一回事了。
- 提交总是会成功的，因为提交是在本地进行的。甚至基于旧版本的改动也可以成功提交，提交会基于旧的版本创建一个新的分支。
- 不再像是 SVN 那样搞“提交竞赛”了，而是在需要的时候才进行合并和冲突解决。在多人协作开发时，SVN 的提交竞赛经常会使提交被打断，而 Git 的每个用户就好像工作在独立的 Feature Branch（功能分支）中一样，它的提交不会被打断，直到你的工作完全满意为止，你可以 PUSH 给他人或者他人 PULL 你的版本库。
- 合并会发生在 PULL 和 PUSH 的过程中，不能自动解决的冲突会提示您手动完成。
- 可以为 Git 版本库进行授权，比如谁能创建版本库，谁能向版本库 PUSH，谁能够读取（克隆）版本库。

- 团队的成员先将服务器的版本库克隆到本地，然后从服务器的版本库中 PULL 最新的更新（避免网络缓慢的情况）。
- 团队的成员将自己的改动 PUSH 到服务器的版本库中，当其他人和版本库同步（PULL）时，会自动获取改变（避免网络缓慢的情况）。
- Git 的集中式工作模式非常灵活，你完全可以在脱离 Git 服务器所在网络的情况下（如移动办公/出差时），照常使用代码库。
- 你在能够接入 Git 服务器所在的网络时，只需要 PULL 和 PUSH 即可完成和服务器的同步及提交操作。
- Git 提供 rebase 命令，可以让你的改动看起来是基于最新的代码实现的。

3. 在 FreeBSD8.1 下安装 Git 服务器

在 FreeBSD8.1 下安装 Git 服务器，过程很方便。

(1) 用 ports 安装 Git，命令如下：

```
cd /usr/ports/devel/git
make install clean
```

(2) 修改/etc/rc.conf，让 Git 随开机而启动，如下所示：

```
git_daemon_enable="YES"
```

(3) 新增使用者 Git，如下所示：

```
pw useradd git
```

(4) 启用 git daemon，如下所示：

```
/usr/local/etc/rc.d/git_daemon start
```

(5) 用 sockstat 来验证 Git 是否启动，如下所示：

```
sockstat -4l |grep 9418
root    git-daemon 37064 3  tcp4    192.168.21.248:9418  * :*
```

4. 导入一个新的 Git 项目

(1) 把自己介绍给 Git 系统，比如自己的姓名和 E-mail 地址，命令如下：

```
git config --global user.name "Andrew.yu"
git config --global user.email "yuhongchun027@163.com"
```

(2) 提交我的目录/home/andrewy/test 到 Git 项目库中，命令如下：

```
cd /home/andrewy/test/
git init
git add .
git commit
```

(3) 这时候大家可观察我们的/home/andrewy/test 的目录，如下所示：

```
ls -lsart
total 8
```



```

2 -rw-r--r-- 1 root wheel 18 Apr 7 07:42 3
0 -rw-r--r-- 1 root wheel 0 Apr 7 07:42 2
0 -rw-r--r-- 1 root wheel 0 Apr 7 07:42 1
2 drwxr-xr-x 8 root wheel 512 Apr 7 07:42 .git
2 drwxr-xr-x 3 root wheel 512 Apr 7 07:43 .
2 drwxr-xr-x 4 root wheel 512 Apr 7 07:43 ..

```

git init 命令可用于初始化当前所在目录的这个项目，SHELL 返回的提示表明已经建立了一个 .git 隐藏目录来保存这个项目的进展信息。

5. Git 命令行操作

查看源代码和快照的区别，命令如下：

```
git diff
```

查看快照和仓库的区别，命令如下：

```
git diff --cached
```

查看整体改动的信息，命令如下：

```
git status
```

告诉 Git 我修改了 hello.py 文件，请检查一下（Git 中无论修改还是增加了新文件均会使用 add 进行登记），命令如下：

```
git add hello.py
```

提交到 Git，命令如下：

```
git commit
```

查看日志，命令如下：

```
git log
```

只回退 commit 的信息，但 index file（就是临时存储区域）和修改的代码仍然在，命令如下：

```
git reset --soft HEAD
```

不仅回退 commit 的信息，代码也恢复到修改前的版本，命令如下：

```
git reset --hard HEAD
```

回退 commit 和 index file 的信息，保留代码的修改，命令如下：

```
git reset --mixed HEAD
```

默认情况等同于 mixed，命令如下：

```
git reset HEAD
```

从 index file 中删除一个已经登记的文件，命令如下：

```
git reset --a.py
```

创建 branch-a 分支，命令如下：

```
git branch branch -a
```

切换到 branch-a 分支，命令如下：

```
git checkout branch -a
```

将分支合并到主干（如果有冲突会提示将冲突的地方修改后再合并），命令如下：

```
git merge branch -a
```

分支内容合并到主干后删除 branch-a 分支，命令如下：

```
git branch -d branch -a
```

不论分支内容是否合并到主干上，均删除 branch-a 分支，命令如下：

```
git branch -D branch -a
```

克隆 hello-git 到 hello-git2 中，命令如下：

```
git clone /home/flynewton/hello -git hello -git2
```

cd 进入 hello-git2 目录后，接着进行如下操作：

- 修改 hello-git2 中的源代码并提交。
- 创建分支 branch-b，并修改分支中的源码并提交。
- 进入 hello-git 文件夹中。

然后将 hello-git2 的主干代码作为本地的新分支 hello2-works，命令如下：

```
git fetch /home/flynewton/hello -git2 master:hello2 -works
```

查看本地主干和 hello2-works 分支的差异，命令如下：

```
git -p master..hello2 -works
```

合并 hello-git2 的主干到本地，命令如下：

```
git pull /home/flynewton/hello -git2 master
```

合并 hello-git2 的分支到主干，命令如下：

```
git pull /home/flynewton/hello -git2 branch -b
```

以上即是我们在工作中会用到的关于 Git 的命令行操作。另外要说明的是，Git 操作都是基于命令行的，这里将不对其图形化客户端作说明，有兴趣的朋友请自行研究。

在公司的内网开发环境中，我们的开发部门主要是用 SVN 来进行版本管理的，而 Git 目前处于测试阶段，感觉用它来添加项目的子目录不是特别理想。另外，分支合并时产生的 Merge 问题也让人很烦；相反，SVN 在这些方面都做得很好，而且公司毕竟是集中式的版本管理工作模式，所以 SVN 相对来说更合适。关于版本管理软件的选择，我建议大家根据公司的情况来定，不一定非要二选一，有其他的选择也可以，比如很多公司喜欢用收费的 Perforce。该软件功能也比较强，而且是跨平台的，支持 Windows、Linux/Unix、Mac，还有 cygwin 版本的，大家如有需要也可以尝试使用。

2.5 在 FreeBSD8.1 下搭建 Samba 文件服务器

2.5.1 Samba 概述

1. SMB 协议

SMB (Server Message Block, 服务信息块) 协议可以看做是局域网上共享文件/打印机的一种协议, 它可以为网络内部的其他 Windows 和 Linux 机器提供文件系统、打印服务或是其他的一些信息。SMB 的工作原理是让 NetBIOS (Win95 网络邻居通信协议) 与 SMB 这两种协议运行在 TCP/IP 的通信协议上, 且使用 NetBIOS nameserver 让用户的 Linux 机器可以在 Windows 的网络邻居里看到, 然后可以和 Win95/NT 主机在网络上相互沟通, 共享文件与服务。

2. Samba

Samba 是用来实现 SMB 的一种软件, 由澳大利亚的 Andrew Tridgell 开发, 是一种在 Linux (Unix) 环境下运行的免费软件。

3. Samba 可以实现的功能

通过使用 Samba, Linux/Unix 系统可以实现如下功能:

- ☐ 文件服务和打印服务 (在 Linux 和 Win95/NT 系统之间提供打印机和磁盘的共享), 这点在日常工作中使用得比较多。
- ☐ 可实现主要域控制器和域中成员的功能。
- ☐ WINS 服务器及浏览的功能。
- ☐ 支持 SSL (Secure Socket Layer)。
- ☐ 支持 SWAT (Samba Web Administration Tool)。
- ☐ Samba 除了支持 Linux (Unix) 和 Win95/NT 之外, 还支持 DOS、IBM OS/2、Macintosh 等操作系统。

2.5.2 在 FreeBSD8.1 下安装配置 Samba3.4

FreeBSD8.1 下安装 Samba3.4 很方便, 步骤如下所示。

(1) 用 port 安装 Samba3.4, 如下所示:

```
cd /usr/ports/net/samba3.4
make install clean
```

(2) 采用默认配置, 不进行任何改动, Samba 即可工作, 命令如下:

```
cp /usr/local/share/examples/samba34/smb.conf.default /usr/local/etc/smb.conf
```

(3) 添加 smbpasswd 用户, 命令如下:

```
/usr/local/bin/smbpasswd -a username
```

添加用户名, 根据提示信息来进行操作。

(4) 将 Samba 配置成开机即可启动。

我们用 vim 编辑 /etc/rc.conf 文件, 加入如下内容:

```
nmbd_enable="YES"
smbd_enable="YES"
```

手动启动 Samba 服务，命令如下：

```
/usr/local/etc/rc.d/samba start
```

(5) 用 `sockstat` 命令来验证 Samba 是否能正常启动，命令如下：

```
sockstat -4l
root      smbd      2319  22 tcp4  192.168.21.248:445  * : *
root      smbd      2319  23 tcp4  192.168.21.248:139  * : *
root      nmbd      2313   9 udp4  192.168.21.248:137  * : *
root      nmbd      2313  10 udp4  192.168.21.248:138  * : *
root      nmbd      2313  11 udp4  192.168.21.248:137  * : *
root      nmbd      2313  12 udp4  192.168.21.248:138  * : *
```

2.5.3 Samba 的详细语法配置

1. Samba 的语法结构

在安装完 Samba 后，还需要定制它的配置文件 `smb.conf`，才能使 Samba 正常工作以符合要求。由于 SMB 是一个非常复杂的协议，所以配置 Samba 的工作也比较繁琐，大约有 269 条配置项出现在 `smb.conf` 文件中。

`smb.conf` 文件有一个清晰的语法结构，与 Windows 的 `*.ini` 文件十分类似。该文件被分成了几部分，每一部分都包括几个参数，用来定义 Samba 输出的共享信息及详细操作。

文件被分隔成了若干节，每一节都由一个被方括号括起来的标识开始（例如，`[global]`、`[home]`、`[printers]`），每一个配置参数或者是一个全局参数（影响或控制整个服务器），或者是一个服务参数（影响或控制服务器提供的某项服务）。

`global` 部分定义参数用来控制 Samba 的总特性。除 `global` 部分外，每一部分都定义了一个专门的服务。你可以使用下面的语句来指定一个参数：

```
name = VALUE
```

`name` 可以是一个单词或用空格隔开的多个单词。`VALUE` 可以是布尔值（`true` 或 `false`；`yes` 或 `no`；`1` 或 `0`）、数字或字符串。注释以分号开头，可以单独一行，也可以跟在一条语句之后。通过在一行的最后一个字符后加反斜杠“`\`”可以将一行分成多行。每一部分的名字和参数都不区分大小写，例如，参数 `browseable = yes` 与 `browseable = YES` 是完全等价的。

2. smb.conf 文件的功能

`smbd` 和 `nmbd` 这两个守护进程启动时（通常为系统引导时）读配置文件 `smb.conf`，这一配置文件向这两个守护进程说明输出什么共享、共享输出给谁，以及如何进行输出等。因为安全是最重要的，所以你必须指定哪些计算机可以访问这一共享，`smb.conf` 文件可以很灵活地明确指定每一服务都有哪些用户可以访问。随着 Linux 网络的增大，这一控制越来越重要。

3. smb.conf 文件结构

`smb.conf` 文件主要包括以下 3 部分。

- global: (全局) 参数
- directory shares: (目录共享) 部分, 包括标准的 [home] 部分
- printer shares: (打印共享) 部分

其中, global 参数用来设置整个系统的规则; [home] 部分和 [printer] 部分是服务的特定例程, services (服务) 在这里是 Samba 的专用术语, 这些服务定义了哪些用户可以访问这些目录和打印机, 以及如何访问它们。

下面的清单给出了 smb.conf 文件一个简单的例子, 如下所示:

```
[global]
workgroup = MYGROUP
server string = Samba Server
security = user
printing = lprng
log file = /var/log/samba
lock directory = /var/lock/samba
[homes]
comment = Home Directory
browseable = no
read only = no
[printers]
browseable = no
guest ok = yes
printable = yes
```

在上面的代码中, 在 [global] 段中设置了主机名称, 以及主机所在的工作组名称和浏览时可看到的对本机的描述。安全参数告诉 Samba 使用“用户级别”的安全保护方式。Samba 有两种安全模式: 一是共享级别, 将资源加密码控制; 二是用户级别, 可以使用某一用户的所有资源。建议大家使用用户级别的安全控制, 这个较前者更为方便。

[global] 段中还定义了日志文件目录和锁定文件的位置。日志文件在解决故障和完善系统时是很有用的, 锁定文件可以阻止多个用户同时修改相同的文件。

[homes] 段中的设置控制了每一个用户主目录的共享权限。comment 参数指定的字符串在你浏览本机资源时出现在指定资源的旁边。

browseable 参数控制一项服务是否能够出现在网络资源的浏览表中。browseable = no 意味着这个目录将在浏览时显示为要验证的用户名称。举例来说, 如果指定 browseable = no, 当我浏览这个 Samba 服务器时, 我将看到一个名称为 cuckoo 的共享目录。当指定 browseable = yes 时, 我将看到一个名为 homes 和 cuckoo 的共享目录。设置 read only = no 将允许通过验证的用户对主目录写入。但是, 如果他们主目录的 Unix 权限不允许写入, 那么他们就没有写的权限。无论 Unix 的权限怎样, 设置 read only = yes 后, 他们的主目录是只读的。

printing 命令描述了本地打印系统类型, 这可以让 Samba 知道怎样提交打印任务, 显示打印队列, 删除打印任务和进行其他操作。

如果打印系统是 Samba 所不知道的, 你必须在每次执行打印操作时指明命令。

Samba 配置清单中 [printing] 这一段配置允许任何能够登录到 Samba 服务器的用户使用 printcap 中出现的每一台打印机。在正常情况下, 如果使用用户级别的安全控制, guest ok = yes 并不能

授权每一个用户（使用系统），每一个打印服务必须定义为 `printable = yes`。

2.5.4 Samba 在工作中的总结

1. 在 Windows 下更换用户访问 Samba 服务器

在 Windows 客户机访问 Samba 服务器时，常出现的一种现象是：在建立了访问 Samba 服务器的连接之后，再次访问该服务器时，不再出现身份认证对话框，这样便无法更换用户身份了。很多人喜欢用注销来解决这个问题，这样很麻烦而且也浪费时间。其实造成这一现象的原因是 Windows 本身的机制问题，更确切地说是 SMB 服务的问题。由于 NETBIOS 服务是面向连接的，当客户与 Samba 服务器建立连接时，此连接在一段时间内始终是活跃的，所以当用户再次访问该服务器时，便采用了前面的身份而无须再次验证身份了。解法方法的命令如下：

```
net use \\Samba - Server_IP\IPC $/delete
```

在成功删除共享连接后即可更换用户身份访问 Samba 服务器了。

2. Samba 服务使用的端口

在实验过程中，我发现 Samba 服务虽然监听了 4 个端口，但只有 445 端口是用来传输数据的。为了证实此事，我特地开了防火墙及抓包工具来验证，首先在 Samba 服务器上开启了 iptables，并且将 INPUT 和 OUTPUT 默认设置为 DROP，然后再开放了 TCP 的 445 端口，代码如下所示：

```
iptables -A INPUT -p tcp -s 192.168.0.0/24 --dport 445 -j ACCEPT
```

Samba 服务器本来是不能提供数据服务的，但开放了 TCP 的 445 端口以后，它就能与客户端正常通信并提供数据了。我特地用抓包工具 Wireshark 抓了下包，证实了此说法。很多教材上面说还要采用 137、138、139 等端口，这种说法也是成立的，大家可以看看 Samba 服务使用端口的详细情况，如下所示：

```
* Port 137 (UDP) - NetBIOS name service and nmbd
* Port 138 (UDP) - NetBIOS datagram service
* Port 139 (TCP) - File and printer sharing and smbd
* Port 389 (TCP) - for LDAP (Active Directory Mode)
* Port 445 (TCP) - NetBIOS was moved to 445 after 2000 and beyond (CIFS)
* Port 901 (TCP) - for SWAT
```

我们通过 Wireshark 或 TCPDump 等抓包工具对 Samba 的数据包进行分析，发现：如果 Samba 只进行数据传输的话，445 端口够用了，所以我们只需要在 iptables 防火墙上打开 445 端口。

3. 通过主机名称和 IP 地址列表控制 Samba 访问

在 FreeBSD8.1 中 Samba 的配置文件是 `/usr/local/etc/smb.conf`，我们可以编辑 `/usr/local/etc/smb.conf` 文件，在 `[global]` 项中添加如下代码：

```
hosts allow = 192.168.1.0/24 127.0.0.1
```

即指示 Samba 服务器只接受网络 192.168.1.0 中的主机和本机访问，也可以使用主机名代替 IP 地址，如果无此项的话，即默认允许所有主机访问。加入此项后，就默认禁止 192.168.1.0 和本机以外的主机访问 Samba 服务器了，代码如下所示：

```
hosts allow=mail.test.com
```

当然，如果禁止的只是少数机器可以用下列语法来实现：

```
hosts allow=test.com EXCEPT mail.test.com
```

在实际工作中，我推荐用点分十进制的方法来记忆：即 `hosts allow = 192.168.1.0/255.255.255.0`。事实上，点分十进制在 Linux 的其他应用服务是一种通用的写法，有兴趣的朋友可以做实验验证。如果在现实中要同时实现用户服务和主机的访问控制，可采用 PAM（可插入认证模块），它所实现的功能是允许特定用户通过特定主机访问 Samba 服务器，这不是平时工作的重点，不再详细叙述，有兴趣的朋友可自行研究。

2.5.5 Linux 下的高级权限文件控制

在企业内网的开发环境方面，文件服务器是一个非常重要的环节。其中，Samba 服务器由于其权限控制的高度灵活性，最初学习时确实会让大家感到很迷惑，但我们可以先搭建一些简单的案例来掌握其语法。本节主要是介绍 Linux 下的高级权限 `suid`、`sgid`、`sticky` 三种权限的特点。

很多人都很奇怪，为什么我们需要学习这三种权限呢？因为在实际工作中我们发现，如果不了解这些特殊权限会让我们对 Linux 权限的理解（尤其是加上 Samba 权限后）尤为困难，所以我们必须要学习并了解它们。下面试图用浅显的讲解，让大家能充分理解这三种权限的作用。理解了它们，再理解 Samba 的权限控制就更容易了。

注意 Samba 的权限由两方面构成：一是目录本身的权限；二是 Samba 的配置权限。最终权限的定义是两者的最小交集。

我们接着来理解这 3 种权限：

- 1) 一个文件都有一个所有者，表示该文件是谁创建的。
- 2) 如果同时该文件还有一个组编号，则表示该文件所属的组一般为文件所有者所属的组。
- 3) 如果是一个可执行文件，那么在执行时，一般该文件只拥有调用该文件的用户所具有的权限。

权限标志通过以下 3 个“位”来定义。

- `setuid`：设置使文件在执行阶段具有文件所有者的权限。比如 `/usr/bin/passwd`，如果是一般用户执行该文件，则在执行过程中，用户通过该文件可以获得 `root` 权限，从而可以更改用户的密码。
- `setgid`：该权限只对目录有效。目录被设置该权限后，任何用户在此目录下创建的文件都具有和该目录所属的组相同的组。
- `sticky`：该位可以理解为防删除位。一个文件是否可以被某用户删除，主要取决于该文件所属的组是否对该用户具有写权限。如果没有写权限，则这个目录下的所有文件都不能被删除，同时也不能添加新的文件。如果希望用户能够添加文件但同时又不删除文件，则可以对文件使用 `sticky bit` 位。设置该位后，就算用户对目录具有写权限也不能删除该文件。

下面介绍一下三种权限的特点。

(1) sticky 的特点

- sticky 只能应用在目录上，并且是应用在其他人的目录上。
- 只有 root 和文件的拥有者才能删除该文件。
- 小写 s 表示能执行，大写 S 表示不能执行。

它的其他特点大家可以参考/tmp 目录。

(2) setuid 的特点

- setuid 只能应用在二进制文件中。
- 当一个文件应用了 setuid，那么任何人在执行该命令的时候就能临时拥有该文件拥有者的权限。
- setuid 只能应用在文件的拥有者上。
- 小写 s 表示能执行，大写 S 表示不能执行。

它的其他特点大家可以参考/usr/bin/passwd 目录。

(3) setgid 的特点

- 既可以应用在文件上，也可以应用在目录上。
- 当 setgid 应用在目录上时，任何人在该目录中建立的文件和目录的拥有者属于目录所属组。
- 应用在拥有组上。
- setgid 应用在文件上时，任何人在执行该文件时，临时拥有该文件所属组的权限。
- 小写 t 表示可执行，大写 T 表示不能执行。
- 可应用于同组开发人员的共用资料上，保证安全。

注意 大小写问题没明白的朋友请不要着急，后面的内容会讲到这个问题。

如何操作带有这些标志的文件呢？

操作这些标志与操作文件权限的命令是一样的，都是 chmod。有两种方法来操作。

第一种方法如下：

```
chmod u+s temp
```

表示为 temp 文件加上 setuid 标志（setuid 只对文件有效）。

```
chmod g+s tempdir
```

表示为 tempdir 目录加上 setgid 标志（setgid 对目录和文件有效）。

```
chmod o+t temp
```

表示为 temp 文件加上 sticky 标志（sticky 只对文件有效）。

第二种方法是采用八进制方式。一般文件是通过 3 组八进制数字来设置标志的，如 666、777、644 等。如果设置了这些特殊标志，则在这组数字之外再加一组八进制数字，如 4666、2777 等。在八进制的数字中三位数字（我们这里以为 abc 来举例）的意义分别如下：

- a 为 setuid 位。如果该位为 1，则表示设置 setuid。
- b 为 setgid 位。如果该位为 1，则表示设置 setgid。
- c 为 sticky 位。如果该位为 1，则表示设置 sticky。

我习惯用第一种方法来操作，但许多时候文件的权限表示都是用数字来执行的。所以，建议这两种方法都掌握。

设置完这些标志以后，可以用 `ls -lsart` 来查看。如果有这些标志，则会在原来的执行标志位置上显示。如：

- ☐ `rwsrw-r--` 表示有 `setuid` 标志。
- ☐ `rw-rwsrw-` 表示有 `setgid` 标志。
- ☐ `rw-rw-rwt` 表示有 `sticky` 标志。

那么原来的执行标志 `x` 到哪里去了呢？系统是这样规定的，如果本来在该位上有 `x`，则这些特殊标志显示为小写字母 (`s`、`s`、`t`)；否则，显示为大写字母 (`S`、`S`、`T`)。

这 3 个权限的数字位可以如下理解（这是我的理解和记忆参考法，仅供大家参考学习）：

```
[root@ server3 test]# 1 1 1
[root@ server3 test]# rws rws rwt
[root@ server3 test]#
[root@ server3 test]# suid sgid sticky
```

所以，可以得出：

- ☐ `chmod 4777` 是设置 `setuid`。
- ☐ `chmod 2777` 是设置 `setgid`。
- ☐ `chmod 1777` 是设置 `sticky`。

常用操作如下所示。

- ☐ 找出所有危险的目录（设置目录所有人可读写却没有设置 `sticky` 位的目录），命令如下：

```
find / -perm -0007 -type d
```

- ☐ 找出所有设置了 `suid` 的文件，代码如下：

```
find / -perm -4000 -type f
```

2.5.6 Samba 在企业开发环境中的常用案例之一

我们的开发服务器上有所有开发人员的账号，大家主要是在自己的 `home` 目录里 checkout SVN 服务器的代码从而进行开发，备份服务器每天都会对大家各自的 `home` 目录进行备份。由于我们的办公环境都还是 Windows 系列的，这样就存在着一个与 Samba 服务器交互的问题。以前我们用 Xbrowser 和 Winscp，但大家都反映它们的速度过慢，所以我们将开发人员均添加到 Samba 用户库中，每个人都可以默认对自己的 `/home/` 用户目录进行读写。在 Windows 下我们可以用磁盘映射的办法来操作，这样 Windows 机器默认启动后，就有一个远程的磁盘可以读写，能方便自己工作。另外通过测试我们也发现，Samba 的速度算是非常快的（特别是相对于 Xbrowser 和 Winscp 而言），而且我们还可以将此盘作为文件中转站，效果图如图 2-50 和图 2-51 所示。

在 Windows XP 下挂载 Samba 的文件磁盘是我在日常工作中最常用的一种做法，我们可以将 Samba 用户目录作为我们的一个 checkout，挂到 Windows 下面用工具来查看和编辑，最好是直接映射成磁盘（因为 Samba 文件是不重启和关机的，这样就相当于我们又有了一块磁盘可以调用，最关键的是这个磁盘还是开发服务器下的），这样工作起来就方便多了。

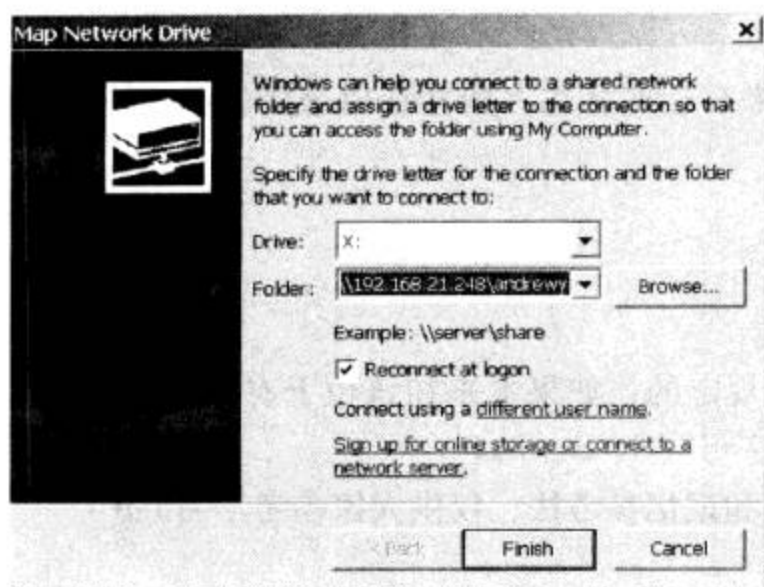


图 2-50 在 Windows 下映射 Samba 磁盘



图 2-51 在 Windows 下挂载 Samba 用户目录

2.5.7 Samba 在企业开发环境中的应用案例之二

之前我们介绍了 Samba 的实用技巧与三种权限的机制，下面介绍一个 Samba 的企业应用案例。本节介绍的 Samba 应用案例在很多公司都可能用得上，只要根据具体情况修改配置文件就可以了。其实原理也很简单，就是用户的读与写权限要分开。

1. 案例要求与分析

需求 1

所有与员工都能在公司内漫游办公，但不管是在哪台计算机上工作，都要把自己的文件数据库存在 Samba 文件服务器上。

分析 1:

需要将 Samba 作为文件服务器，为所有的用户创建账号和目录。

需求 2

- 为所有的用户创建账号和目录，不分配 shell。
- 假设市场部的 Tom、Jack 属于 sales 组。
- 技术部的 Red、Blue 属于 tech 组。
- 总经理是 CEO，财务是 Fineance，他们均属于 lead 组。

分析 2:

- 建组 sales、tech 及 leader 组。
- 所有市场部的员工加入 sales 组。
- 技术部的员工加入 tech 组。
- CEO 和 Fineance 加入 leader 组。

需求 3

sales 组的文件只有 Tom 和 leader 组的人可以读写，其他人只能看（非认证用户是不能进入目录的）。

tech 组的文件只有 Red 和 leader 组的人可以读写，其他人只能看（非认证用户是不能进入目录的）。

分析3:

通过 Samba 权限控制来达成此目的。

2. 实施步骤

(1) 启动 Samba, 命令如下:

```
/usr/local/etc/rc.d/samba start
```

先建立组 and 用户, 把用户加入到相应的组中, 给用户分配一个不可用的 shell。由于组比较多, 为了避免混淆, 我建议大家纸上画好组织结构图, 这样也方便自己理解。

(2) 添加用户组, 命令如下所示:

```
pw groupadd sales
pw groupadd tech
pw groupadd leader
```

然后用 `pw groupshow` 了解到 sales、tech 及 leader 的 gid 分别为 1005、1006 和 1007。

添加用户, 命令如下所示:

```
pw useradd tom -s /sbin/nologin -g 1005
pw useradd jack -s /sbin/nologin -g 1005
pw useradd red -s /sbin/nologin -g 1006
pw useradd blue -s /sbin/nologin -g 1006
pw useradd ceo -s /sbin/nologin -g 1007
pw useradd fineance -s /sbin/nologin -g 1007
```

将其分别添加至 Samba 用户库, 此处的 Samba 密码尽量选择不一样的, 并注意密码的复杂性。命令如下所示:

```
smbpasswd -a jack
New SMB password:
Retype new SMB password:
Added user jack.
smbpasswd -a red
New SMB password:
Retype new SMB password:
Added user red.
smbpasswd -a blue
New SMB password:
Retype new SMB password:
Added user blue.
smbpasswd -a ceo
New SMB password:
Retype new SMB password:
Added user ceo.
smbpasswd -a fineance
New SMB password:
Retype new SMB password:
Add user fineance.
```

(3) 建立 Samba 的文件目录, 并分配权限, 命令如下:

```
mkdir /home/sales
mkdir /home/tech
```

分别给这两个目录权限，命令如下：

```
chmod 777 /home/sales
chmod 777 /home/tech
```

给予 sales 和 tech 目录 777 权限的目的是，为了防止它们被本身的权限控制，用户能访问目录的最终权限是被文件目录的权限和 Samba 权限的最小权限交集所控制的。

```
chgrp sales /home/sales
chgrp tech /home/tech
chmod g+s /home/sales
chmod g+s /home/tech
```

给这两个目录分配 sgid 权限，将能建立的文件都划归其组所有，命令如下：

```
ls -ld /home/sales /home/tech ls -ld /home/sales /home/tech
```

我们可以用命令 ls -ld 来查看目录的属性，命令如下：

```
ls -ld /home/sales /home/tech
drwxrws --- 2 root sales 512 Jan 25 05:36 /home/sales
drwxrws --- 2 root tech 512 Jan 25 05:36 /home/tech
```

(4) 用 Samba 的权限来严格限制这两个目录的权限。修改配置文件 vim/etc/samba/smb.conf，在文件的最后加入如下代码：

```
[sales]
path = /home/sales
public = no
valid users = @ sales, @ leader
write list = tom, @ leader
create mask = 0770
directory mask = 0770

[tech]
path = /home/tech
public = no
valid users = @ tech, @ leader
write list = red, @ leader
create mask = 0770
directory mask = 0770
```

(5) 重启服务器，命令如下：

```
/usr/local/etc/rc.d/samba start
```

3. 总结

测试时我为了避开 Windows XP 的 NETBIOS 的 Bug，直接开了 3 台 Windows XP 机器，外加 n 台虚拟机。测试结果还是比较成功的，其实这里主要也只牵涉到 3 个知识点，归纳如下：

□ 用户的最终权限分配为目录的权限和 Samba 权限的最小交集。

- valid 为认证用户，没经过认证的用户是不能看到相应目录的文件的。
- write list 会覆盖用户原有的用户权限，即使他们原先只有读的权限。

无论是在我们的内网开发环境上，还是在线上服务器（用于与线上的 FreeBSD 文件服务器与 Windows 2003 服务器交互）上 Samba 服务让我们的工作更为便利，而且其稳定性也是让人非常放心的。当然了，Samba 也有其更高级的用法，比如用其作域控或与 Windows 2003 下的域控相结合，但内网环境我们本着方便的原则没有采纳这种做法，这里就不介绍了，有兴趣的朋友可自行研究。

2.6 在 FreeBSD8.1 下配置 NFS 文件服务器

网络文件系统是 FreeBSD 所支持的文件系统中的一种，也被称为 NFS。NFS 允许一个系统在网络上与他人共享目录和文件。通过使用 NFS，用户和程序可以像访问本地文件一样访问远端系统上的文件。NFS 在 Linux/Unix 下是一个比较重要的服务，建议大家掌握。

1. NFS 文件系统的好处

- 本地工作站使用更少的磁盘空间，因为通常的数据可以存放在一台存储服务器上而且可以通过网络访问到。
- 用户不必在每个网络上机器里头都有一个 home 目录。home 目录可以被放在 NFS 服务器上并且在网络上处处可用。
- 诸如软驱 CDROM 和 Zip（是指一种高储存密度的磁盘驱动器与磁盘）之类的存储设备可以在网络上面被别的机器使用，这可以减少整个网络上的可移动介质设备的数量。

2. NFS 的实际应用

- 多机组建负载均衡集群时，可以使用 NFS 共享存储，保证各服务器读写文件的一致性。
- 多个机器共享一台 CDROM 或其他设备。这对于在多台机器中安装软件来说更加便宜且方便。
- 在大型网络中，配置一台中心 NFS 服务器来放置所有用户的 home 目录可能会带来便利。这些目录能被输出到网络中，不管用户在哪台工作站上登录，总能得到相同的 home 目录。
- 几台机器可以有通用的 /usr/ports/distfiles 目录。这样的话，当你需要在几台机器上安装 port 时，则无须在每台设备上下载而且可以快速访问源码。

注意 NFS 由 Sun microsystems 公司开发。它是一种网络操作系统，并且是 Unix 操作系统的协议。

3. NFS 的工作方式

NFS 至少包括两个主要的部分：一台服务器，以及至少一台客户机，客户机远程地访问保存在服务器上的数据。要让这一切运转起来，需要配置并运行下面几个服务。

- nfsd：来自 NFS 客户端的请求服务。
- mountd：NFS 挂载服务，处理 nfsd 递交过来的请求。
- rpcbind：此服务允许 NFS 客户程序查询正在被 NFS 服务使用的端口。

客户端同样会运行一些进程，比如 nfsiod，nfsiod 会处理来自 NFS 的请求。

下面介绍一下 NFS 的安装配置过程。此过程比较简单，我们可以按照以下步骤操作，这里为了方便操作，我切换到了 root 下操作。

(1) 我们在 NFS 服务端安装 `rpc`，命令如下：

```
cd /usr/ports/math/rpc
sudo make install clean
```

确认 `/etc/rc.conf` 文件里头以下开关都配上了，如果此文件没有如下内容，则添加：

```
nfs_server_enable="YES"
rpcbind_enable="YES"
nfs_server_flags="-u -t -n 4"
mountd_flags="-r"
```

每当 NFS 服务器启动时，`mountd` 就会自动运行。在选项 `nfs_server_flags` 中，`-u` 表示我们提供 UDP 方式的联机，`-t` 表示以 TCP 方式联机，`-n 4` 是告诉 `nfsd` 运行自己的 4 个相同的拷贝。

注意 某些应用程序需要文件上锁支持才能正常运行。在使用 NFS 文件系统时，可以用 `rpc.lockd` 来支持文件上锁功能。要启用它，需要在服务器和客户机的 `/etc/rc.conf` 中加入以下内容（假定两端都是 FreeBSD 系统，并且均已配好了 NFS）：

```
rpc_lockd_enable="YES"
rpc_statd_enable="YES"
```

然后使用下述命令启动该程序，如下所示：

```
/etc/rc.d/lockd start
/etc/rc.d/statd start
```

(2) 客户端的配置。在客户端一侧，确认下面这个开关出现在 `/etc/rc.conf` 里头，如下所示：

```
nfs_client_enable="YES"
nfs_client_flags="-n 4"
```

(3) `exports` 配置文件的相关设置参数，此项配置是在 NFS 服务器端配置的。

`/etc/exports` 文件指定了 NFS 服务器端共享了哪个目录，在 `/etc/exports` 里面，每行指定一个输出的文件系统和哪些机器可以访问该文件系统。在指定机器访问权限的同时，访问选项开关也可以被指定。

下面给出实例说明，我们可以用 `cat` 或 `vim` 编辑或查看 `/etc/exports` 文件。

```
cat /etc/exports
/usr/src /usr/obj -ro 192.168.9.224
```

`-ro` 表示 read only，只读。

```
/home -alldirs 192.168.9.220 192.168.9.221 192.168.9.222
```

`-alldirs` 标记允许子目录被作为挂载点，但前提是 `/data` 必须是一个独立的 filesystem。

```
# /a -maproot=root -network 192.168.9.0 -mask 255.255.248.0
```

`-maproot=root` 标记授权远端系统上的 `root` 用户在被输出的文件系统上以 `root` 身份进行读写，`-network IP-mask MASK` 指定允许联机的网域。

```
# /cdrom -ro -mapall=alex
```

-mapall 将所有 Client 的存取联机对应到 user 上，也就是说所有人的身份都转成 user。
下面我们依照实例来配置文件/etc/exports，内容如下：

```
#cat /etc/exports
/data -alldirs 192.168.9.223
```

上面表示只允许 IP 为 192.168.9.223 的机器挂载本地的/data 目录。

(4) NFS 的启动与重载配置。

启动 NFS，命令如下：

```
sudo /etc/rc.d/nfsd start
```

重新加载 NFS 配置文件的过程如下：在修改了/etc/exports 文件之后，就必须让 mountd 服务重新检查它，以便使修改生效。一种方法是通过给正在运行的服务程序发送 HUP 信号来完成，命令如下：

```
kill -HUP 'cat /var/run/mountd.pid'
```

另一种方法是指定适当的参数来运行 mountd 脚本，命令如下：

```
/etc/rc.d/mountd reload
```

如果是第一次设定，连 mountd 都还没有启动，那么你可以选择重新开机或是执行下列指令来启动 NFS 服务：

```
rpcbind
nfsd -u -t -n 4
mountd -r
```

NFS 客户端启动 NFS 文件系统服务，命令如下：

```
nfsiod -n 4
```

(5) NFS 服务器正常启动后，我们就可以在客户机上挂载服务器的共享文件了。
查看 NFS 服务器的输出，命令如下：

```
showmount -e 192.168.9.134
```

挂载 NFS 服务器中的共享目录到本地目录/data2/下，命令如下：

```
mount -t nfs 192.168.9.134:/data /data2/
```

卸载系统中已挂载的 NFS 共享目录，命令如下：

```
umount /data/
```

修改 fstab 文件让系统启动时自动挂载 NFS 文件，命令如下：

```
vim /etc/fstab
```

添加以下内容到文件末尾：

```
192.168.1.134:/data /data2 nfs defaults 0 0
```

NFS 文件系统是大家比较熟悉的一个应用服务，许多朋友都能在各种 Linux 版本下非常熟练地配置，这里就不浪费篇幅了。大家要注意区分一下 Centos5.5 下与 FreeBSD8.1 下配置 NFS 文件系统的不同之处。

2.7 在 FreeBSD8.1 与 Centos5.5 下搭建 rsync 服务器

2.7.1 rsync 的概念

rsync (remote synchronize) 是一个远程数据同步工具，可通过 LAN/WAN 快速同步多台主机间的文件。rsync 使用所谓的“rsync 算法”来使本地主机和远程主机之间的文件达到同步，这个算法并不是每次都整份传送，它只传送两个文件的不同部分，因此速度相当快。

rsync 的优点如下：

- 可以镜像保存整个目录树和文件系统。
- 可以很容易地做到保持原来文件的权限、时间、软硬链接等。
- 无须特殊权限即可安装。
- 拥有优化的流程，文件传输效率高。
- 可以使用 rsh、SSH 等方式来传输文件，当然也可以直接通过 socket 连接。
- 支持匿名传输。

另外，与 scp 相比，传输速度不是一个数量级的。我们在局域网时经常用 rsync 和 scp 传输大量 MySQL 数据库文件，发现 rsync 至少比 scp 快 20 倍以上。所以大家如果需要在 Linux/Unix 服务器之间互传海量数据，rsync 是最好的选择。

2.7.2 在 Centos5.5 下配置 rsync 服务器

具体安装步骤如下：

首先准备一台 Centos5.5，系统为 64 位 Centos、IP 为 192.168.21.41，将其作为 rsync 服务器。另外准备一台 64-bit FreeBSD8.1、IP 为 192.168.21.44 的机器作为其客户端。

具体的安装步骤不多说，只介绍重点内容。首先检查 rsync 是否安装，命令如下：

```
rpm -q rsync
rsync-2.6.8-3.1
```

上面的结果说明 rsync 已安装，如果出现 package rsync is not installed 提示，则说明这个软件包没有安装，大家可以使用如下命令进行安装：

```
yum -y install rsync
```

另外，关闭防火墙和 SELinux，因为是在内网中传输，所以没必要打开。关闭它们，免得引起不必要的麻烦，命令如下：

```
service iptables stop
chkconfig iptables off
setenforce 0
```

下面分享一下我自己定义的配置文件/etc/rsyncd.conf（说明：此文件并不是系统创建的，你也

可以给它取不同的名字)。先给出具体代码, 后面再进行详细注释, 代码如下:

```
uid=nobody
gid=nobody
user chroot=no
max connections=200
timeout=600
pid file=/var/run/rsyncd.pid
lock file=/var/run/rsyncd.lock
log file=/var/log/rsyncd.log
[backup]
path=/backup/
ignore errors
read only=no
list=no
hosts allow=192.168.21.0/255.255.255.0
auth users=test
secrets file=/etc/rsyncd.password
```

下面说明一下/etc/rsyncd.conf 的语法。

```
uid=nobody
```

上面指的是进行备份的用户, nobody 为任何用户。

```
gid=nobody
```

表示进行备份的组, nobody 为任意组。

```
use chroot=no
```

如果 use chroot 指定为 true, 那么 rsync 在传输文件以前首先 chroot 到 path 参数所指定的目录下。这样做可实现额外的安全防护功能, 但缺点是需要给用户 root 权限, 并且不能备份通过外部的符号连接所指向的目录文件。在默认情况下, chroot 值为 true, 但是这一般不需要, 可选择 no 或 false。

```
list=no
```

表示不允许列清单。

```
max connections=200
```

表示最大连接数。

```
timeout=600
```

表示覆盖客户指定的 IP 超时时间, 也就是说 rsync 服务器不会永远等待一个崩溃的客户端。

```
pidfile=/var/run/rsyncd.pid
```

指的是 pid 文件的存放位置。

```
lock file=/var/run/rsync.lock
```

指的是锁文件的存放位置。

```
log file=/var/log/rsyncd.log
```

指的是日志文件的存放位置。

```
[backup]
```

这是认证模块名，即跟 Samba 语法一样，是对外公布的名字。

```
path = /backup/
```

这是参与同步的目录。

```
ignore errors
```

表示可以忽略一些无关的 I/O 错误。

```
read only = no
```

表示允许读和写。

```
list = no
```

表示不允许列清单。

```
hosts allow = 192.168.21.0/255.255.255.0
```

这里跟 Samba 的语法是一样的，只允许 192.168.21.0/24 的网段进行同步，拒绝其他一切网段连接。

```
auth users = test
```

指的是认证的用户名。

```
secrets file = /etc/rsyncd.password
```

指的是密码文件的存放地址。

启动服务端的 rsync，可通过 xinetd 来控制，这里要对 rsync 进行修改，我们先编辑 rsync 相关的文件 `etc/xinetd.d/rsync`，如下所示：

```
service rsync
{
    disable = yes
    socket_type = stream
    wait       = no
    user       = root
    server      = /usr/bin/rsync
    server_args = --daemon
    log_on_failure += USERID
}
```

将 `disable = yes` 改为 `disable = no`，然后重启 xinetd 即可，命令如下：

```
/etc/init.d/xinetd restart
```

配置中应该注意的问题如下。

□ `[backup]`：认证模块名，这个认证模块名是服务器对外的名字，机器同步时只会认这个名字。

这里的 `path` 不要随便设置，要知道这里是认证模块，以后从客户机备份的数据会存储在这里。

- `auth users = redhat`: 认证的用户名。这个名字是服务器端实实在在存在的用户，如果服务器端少了它，估计你的数据同步就无法实现了，请不要忽略它。
- `path = /backup/`: 参与同步的目录的权限。如果此目录权限不够，rsync 同步是成功不了的。这需要稍后在根目录下自己建，并分配相应的写权限，如下所示：

```
mkdir /backup
chmod -R 777 /backup
echo "test:test" > /etc/rsyncd.password
```

说明 这里我设置的是用户名和密码一致。

为了安全起见，我设置他的权限为 600，如下所示：

```
chmod 600 /etc/rsyncd.password
```

客户端配置如下：

```
echo "test" > /etc/rsyncd.password
```

这里只需要密码，不需要用户名，免得要同步时还要手动互动。为了安全，一样配置 600 的权限，如下所示：

```
chmod 600 /etc/rsyncd.password
```

下面说说在工作中经常遇到的 rsync 问题。

故障一：服务器端的目录不存在或无权限。故障描述如下：

```
@ ERROR: chroot failed
rsync error: error starting client - server protocol (code 5) at main.c(1522) [receiver=3.0.3]
```

解决方法：创建目录或修改目录权限。

故障二：服务器端该模块（tee）需要验证用户名和密码，但客户端没有提供正确的用户名和密码，认证失败。故障描述如下：

```
@ ERROR: auth failed on module tee
rsync error: error starting client - server protocol (code 5) at main.c(1522) [receiver=3.0.3]
```

解决方法：提供正确的用户名和密码。

故障三：服务器上不存在指定的模块。故障描述如下：

```
@ ERROR: Unknown module 'tee_nonexists'
rsync error: error starting client - server protocol (code 5) at main.c(1522) [receiver=3.0.3]
```

解决方法：提供正确的模块名。

我们接着可以进行测试工作了。

在 FreeBSD8.1 的机器上执行如下命令：

```
rsync -vzrtopg --delete /home/andrewy/etc test@192.168.21.41::backup --password-file = /etc/rsyncd.password
```

这时候就可以看到正确的同步效果了。

下面再说说工作中经常用到的 rsync 参数。

如果不需要交互式的操作，rsync 平时也可以像 scp 那样工作，我由于经常要在服务器之间拷贝 MySQL 文件，MySQL 的库文件有时有几十 GB，感觉 rsync 在速度上还是占绝对优势的。以下是我在工作中经常用到的 rsync 参数：

- -v --verbose：详细模式输出。
- -r --recursive：对子目录以递归模式处理。
- -p --perms：保持文件权限。
- -o --owner：保持文件属主信息。
- -g --group：保持文件属组信息。
- -t --times：保持文件时间信息。
- --delete：删除那些 DST 中存在而 SRC 中不存在的文件或目录。
- --delete-excluded：同样删除接收端那些被该选项指定排除的文件。
- -z --compress：对备份的文件在传输时进行压缩处理。
- --exclude = PATTERN：指定排除不需要传输的文件模式。
- --include = PATTERN：指定不排除需要传输的文件模式。
- --exclude-from = FILE：排除 FILE 中指定模式的文件，FILE 中的文件路径为相对路径。
- --include-from = FILE：不排除 FILE 指定模式匹配的文件，FILE 中的文件路径为相对路径。

2.7.3 在 FreeBSD8.1 下配置 rsync 服务器

通过前面的介绍，大家应该比较清楚如何在 Centos5.5 下配置 rsync 服务器了。那在 FreeBSD8.1 下又该如何配置呢？其实也是很简单，过程如下所示。

(1) 以 FreeBSD8.1 作为 rsync 服务器（IP 为 192.168.4.194），以 Centos5.5（IP 为 192.168.4.222）作为其客户端。

实现目标：让 Centos5.5 的 Linux 机器 /var/www/website/newg 下的所有图片自动 rsync 到 FreeBSD 8.1 机器的 /usr/data/ 目录下，要求不能输入密码自动同步。

(2) 做好安装前的准备工作。

在 FreeBSD8.1 下创建用户并设置权限等，如下所示：

```
sudo pw useradd admin
sudo passwd admin
sudo mkdir -p /usr/data/
sudo chown admin:admin /usr/data/
```

为了避免同步时出错，/usr/data 最好给予写权限，命令如下：

```
sudo chmod o+w /usr/data/
```

(3) rsync 在 FreeBSD8.1 下的安装方法有两种，如下所示。

安装方法一：用 pkg_add -r -v 直接安装，命令如下：

```
sudo pkg_add -r -v rsync
```

安装方法二：用 ports 安装，命令如下：


```
cd /usr/ports/net/rsync
sudo make install clean
```

(4) 配置 FreeBSD8.1 下的 rsync 服务器端，命令如下：

```
sudo vim /usr/local/etc/rsyncd.conf
```

添加以下内容：

```
[pics]
comment = web server backup
path = /usr/data/
auth users = admin
uid = nobody
gid = nogroup
secrets file = /usr/local/etc/rsyncd.pass
read only = no
```

配置 rsyncd.pass，如下所示：

```
sudo vim /usr/local/etc/rsyncd.pass
```

加入以下内容：

```
admin:admin101 // 认证所需的用户名/密码,建议设置强密码
sudo chmod 600 /usr/local/etc/rsyncd.pass
```

配置 rc.conf，加入以下内容：

```
rsyncd_enable = "YES"
```

启动 rsync 的 daemon 模式，命令如下：

```
sudo vim /usr/local/etc/rc.d/rsyncd.pass
```

修改这一行的内容，使用 IPv4 协议，如下所示：

```
command_args = "-4 --daemon"
```

配置到此就 OK 了，现在尝试启动 rsync，命令如下：

```
sudo /usr/local/etc/rc.d/rsyncd start
```

检查 rsync daemon 启动状态，命令如下：

```
% sudo sockstat -4l | grep rsync
root rsync 586 3 dgram -> /var/run/logpriv
root rsync 586 4 tcp4 * :873 * :*
```

(5) 接下来是 rsync 的客户端配置（这里我是用的 root 身份）。配置 rsyncd.pass，方法如下所示：

```
vim /usr/local/etc/rsyncd.pass
```

加入以下内容，rsyncd.pass 中只有密码内容，即 admin101，这样在 rsyncd.pass 里指定用户名和密码可以让 rsync 在其认证时不用输入用户名，如下所示：

```
chmod 600 /usr/local/etc/rsyncd.pass
```

(6) 现在进行同步测试，命令如下：

```
rsync -vzrtopg --password-file=/usr/local/etc/rsyncd.pass /var/www/website/newg 192.168.4.222::pics
```

很顺利地配置完毕。

(7) 了解如何实现自动同步。

如果要实现自动同步，可以将上面的这行命令保存为一个 sh 文件，比如 backupdata.sh，然后放进 crontab 中，每天 00:01 的时候执行，如下所示：

```
01 00 * * * root /usr/local/bin/bash /data/backup/backupdata.sh
```

最后说明一点，如果我们的服务器经常有海量的小文件需要备份或同步，建议用 rsync3.0 以上的版本，它的速度明显快于 3.0 的版本。如果服务器上没有相对应的版本（Centos5.5 默认的 rsync 版本为 2.6.8），可以采取源码编译安装的方法。

2.7.4 rsync + Inotify 实现数据的实时同步更新

1. rsync 的优点与不足

大家通过上面的描述和实验应该已经了解 rsync 具有安全性高、备份迅速、支持增量备份等优点，通过 rsync 可以解决对实时性要求不高的数据备份需求，例如定期地备份文件服务器数据到远端服务器，对本地磁盘定期做数据镜像等。

随着应用系统规模的不断扩大，对数据的安全性和可靠性也提出了更好的要求，rsync 在高端业务系统中也逐渐暴露出了很多的不足之处：首先，rsync 同步数据时，需要扫描所有文件后进行比对，然后进行增量传输。如果文件数量达到了百万甚至千万量级，扫描所有文件将是非常耗时的。而且正在发生变化的往往是其中很少的一部分，这是非常低效的方式。其次，rsync 不能实时地去监测、同步数据，虽然它可以通过 Linux 守护进程的方式触发同步，但是两次触发动作一定会有时间差，这样就导致了服务端和客户端数据可能出现不一致的情况，无法在应用故障时完全恢复数据。

基于以上原因，考虑采用 rsync + inotify，这样就可以解决这些问题了。

2. 初识 Inotify

Inotify 是一种强大的、细粒度的、异步的文件系统事件监控机制，Linux 内核从 2.6.13 起，加入了对 Inotify 的支持，通过 Inotify 可以监控文件系统中的添加、删除、修改、移动等各种细微事件。利用这个内核接口，第三方软件就可以监控文件系统下文件的各种变化情况，而 Inotify-tools 就是这样一个第三方软件。

在上面章节中，我们讲到，rsync 可以实现触发式的文件同步，但是通过 Crontab 守护进程方式触发，同步的数据和实际数据会有差异，而 Inotify 可以监控文件系统的各种变化，当文件有任何变动时，就触发 rsync 同步，这就刚好解决了同步数据的实时性问题。

3. 安装 inotify 工具 inotify-tools

由于 inotify 特性需要 Linux 内核的支持，在安装 inotify-tools 前要先确认 Linux 系统内核是否达到了 2.6.13 以上，如果 Linux 内核低于 2.6.13 版本，就需要重新编译内核加入对 inotify 的支持，也可以用如下方法来判断内核是否支持 inotify（服务器系统为 Centos5.5 x86_64）：

```
uname -r
2.6.18-194.el5
```

然后通过 ls 来查看是否存在 /proc/sys/fs/inotify 目录，如下所示：

```
ls -lsart /proc/sys/fs/inotify/
总计 0
0 dr-xr-xr-x 7 root root 0 06-16 00:02 ..
0 -rw-r--r-- 1 root root 0 06-21 11:15 max_user_watches
0 -rw-r--r-- 1 root root 0 06-21 11:15 max_user_instances
0 -rw-r--r-- 1 root root 0 06-21 11:15 max_queued_events
0 dr-xr-xr-x 2 root root 0 06-21 11:15 .
```

通过以上显示我们明白，Centos5.5 x86_64 是支持 inotify 的。

4. inotify 可以监控的文件系统事件

Inotify 是文件系统事件监控机制，是 dnotify 的有效替代品（dnotify 是较早内核支持的文件监控机制）。Inotify 是一种强大的、细粒度的、异步的机制，它满足各种各样的文件监控需要，不仅限于安全和性能。

Inotify 可以监视的文件系统事件包括：

- ☐ IN_ACCESS：即文件被访问。
- ☐ IN_MODIFY：文件被 write。
- ☐ IN_ATTRIB：文件属性被修改，如 chmod、chown、touch 等。
- ☐ IN_CLOSE_WRITE：可写文件被 close。
- ☐ IN_CLOSE_NOWRITE：不可写文件被 close。
- ☐ IN_OPEN：文件被 open。
- ☐ IN_MOVED_FROM：文件被移走，如 mv。
- ☐ IN_MOVED_TO：文件被移来，如 mv、cp。
- ☐ IN_CREATE：创建新文件。
- ☐ IN_DELETE：文件被删除，如 rm。
- ☐ IN_DELETE_SELF：自删除，即一个可执行文件在执行时删除自己。
- ☐ IN_MOVE_SELF：自移动，即一个可执行文件在执行时移动自己。
- ☐ IN_UNMOUNT：宿主文件系统被 umount。
- ☐ IN_CLOSE：文件被关闭，等同于（IN_CLOSE_WRITE | IN_CLOSE_NOWRITE）。
- ☐ IN_MOVE：文件被移动，等同于（IN_MOVED_FROM | IN_MOVED_TO）。

注意 上面所说的文件也包括目录。

5. rsync + inotify 企业应用案例

我们公司的后端 Web 是两台部署了 Nginx 的 Web 服务器（如下所示），由于没有共享存储，现在要实现的是对它们的根目录 /data/htdocs/www 实现即时同步更新。

```
WebServer1:192.168.1.5 Centos5.5 x86_64
WebServer2:192.168.1.6 Centos5.5 x86_64
```

根目录均为/data/htdocs/www，自动同步顺序为 WebServer2→WebServer1。我们将 WebServer1 配置成 rsync 的服务器端即可。

(1) 安装 Inotify-tools，可以到 <http://inotify-tools.sourceforge.net/> 下载相应的 inotify-tools 版本，然后开始编译安装。

```
cd /usr/local/src
tar zxvf inotify-tools-3.14.tar.gz
cd inotify-tools-3.14
./configure &&make &&make install
```

(2) WebServer1 端（即 192.168.1.5）的 rsync 配置详见 2.7.2 节的内容。下面是配置好/etc/rsyncd.conf 文件，如下所示：

```
[root@ server ~0m]# vim /etc/rsyncd.conf
uid=nobody
gid=nobody
user chroot=no
max connections=200
timeout=600
pid file=/var/run/rsyncd.pid
lock file=/var/run/rsyncd.lock
log file=/var/log/rsyncd.log

[www]
path=/data/htdocs/www/
ignore errors
read only=no
list=no
hosts allow=192.168.1.0/255.255.255.0
auth users=www
secrets file=/etc/rsyncd.password
```

然后重启 xinetd 即可，如下所示：

```
/etc/init.d/xinetd restart
```

记得两台 Web 机器都要配置/etc/rsyncd.passwd 文件。rsync 的配置过程和原理请大家参考 2.7.2 节的内容，这里就不详细说明了。

注意 为了保证/data/htdocs/www 的目录能够被自动同步，此目录建议分配 777 权限；/etc/rsyncd.password 分配 600，不然实现不了自动同步，同步时会报如下错误：

```
password file must not be other-accessible
continuing without password file
Password:
```

(3) 配置好 WebServer2 的 Inotify 后，写一个 inotify 的监控脚本/root/rsync.sh，脚本内容如下：

```
vim /root/rsync.sh
#!/bin/bash
```



```

src=/data/htdocs/www/
des=www
ip=192.168.1.5

/usr/local/bin/inotifywait -mrq --timefmt '% d/% m/% y % H:% M' --format '% T % w % f' -e modify,delete,create,attrib $src | while read file
do
    rsync -vzrtopg --delete --progress $src www@ $ip::$des --password - file=/etc/rsyncd.password &&
    echo "$src was rsynced"
done

```

脚本相关解释如下：

- --timefmt：指定时间的输出格式。
- --format：指定变化文件的详细信息。

这个脚本的作用就是通过 Inotify 监控文件目录的变化，进而触发 rsync 进行同步操作。由于这个过程是一种主动触发操作，是通过系统内核完成的，所以，比起那些遍历整个目录的扫描方式来，效率要高很多。

然后我们将此脚本放入后台运行，输入如下命令即可：

```
nohup sh/root/rsync.sh&
```

(4) 验证就很容易了，我们可以在 192.168.1.6 的机器的 /data/htdocs/www 目录下新建文件，更改文件内容，可以很欣慰地发现，192.168.1.5 的机器上马上也会发生相应的改变，就像两台机器是网络 RAID1 一样，非常方便。

总体说来，rsync + Inotify 比较适用于没有存储环境的小文件的即时同步更新，如果要更新的文件非常大则不建议用这种方式，如果需要同步的机器数量在 10 台以上时，我建议还是以共享存储的方法来解决。如果没有资金购置昂贵的存储，大家不妨考虑一下 Heartbeat + DRBD + NFS 的架构方案来作为我们的文件服务器。

参考文档如下所示：

<http://ixdba.blog.51cto.com/2895551/580280>

<http://www.ibm.com/developerworks/cn/linux/l-ubuntu-inotify/index.html>

另外，rsync 在 Linux/Unix 下是一个比较重要和实用的服务，它不仅可以用于服务器之间的备份，还很适用做海量小文件（例如多级目录图片）的同步。另外，它同步数据的速度非常快，仅次于 FTP 速度，再加上自身可以配置成无人会话模式，所以许多系统管理员喜欢将其做成 Crontab 计划任务来自动化备份重要服务器资料。由于 rsync 的强大，现在有许多开源软件，都是针对于 rsync 的二次开发，大家可以在 Google 上面了解了一下。建议大家熟悉掌握 rsync 的用法，我们在工作中会经常用到它。

2.8 在 FreeBSD8.1 下搭建 vsftpd 服务器

2.8.1 vsftpd 服务器的特点

vsftpd 是一个基于 GPL 发布的类 Unix 系统上使用的 FTP 服务器软件，它的全称是 Very Secure

FTP，从名称可以看出，编制者的初衷是为了代码安全。除了这与生俱来的安全特性以外，高速与高稳定性也是 vsftpd 的两个重要特点。在速度方面，使用 ASCII 代码的模式下载数据时，vsftpd 的速度是 Wu-FTP 的两倍，如果 Linux 主机使用 2.4 的内核，在千兆以太网上的下载速度可达 86MB/S。vsftpd 在稳定方面就更加出色了，vsftpd 在单机（非集群）上支持 4000 个以上的并发用户同时连接，根据 RedHat 的 FTP 服务器（<http://ftp.redhat.com>）的数据来看，vsftpd 服务器可以支持 15 000 个并发用户。vsftpd 集合高效、易配置、易管理、高安全性于一身，市场应用十分广泛，很多国际性的大公司和自由开源组织都在使用，如：RedHat、Suse、Debian、OpenBSD 等。正是因为它的强大，vsftpd 被作为 RedHat 和 Centos 系列的默认安装组件，在这里我也向大家推荐在 FreeBSD8.1 下采用 vsftpd 作为我们的 FTP 服务器。

2.8.2 vsftpd 的运行模式

vsftpd 运行时有两种模式，一种是独立运行模式，即 standalone 模式；另一种是 xinetd 或 initd 模式，由于在 FreeBSD 下的守护进程是 inetd，所以我们以下都称其为 inetd 模式。

1. standalone 和 initd 模式

像其他守护程序一样，vsftpd 提供了 standalone 和 inetd（inetd 或 xinetd）两种运行模式。简单解释一下，standalone 一次性启动，运行期间一直驻留在内存中，优点是对接入信号反应快，缺点是损耗了一定的系统资源，因此经常会应用于对实时反应要求较高的专业 FTP 服务器。inetd 则恰恰相反，由于只在外部连接发送请求时才会调用 FTP 进程，因此它不适合应用在同时连接数量较多的系统上。此外，inetd 模式不占用系统资源。除了这些以外，inetd 和 standalone 的区别还有：inetd 模式支持 per_IP（单一 IP）限制，而 standalone 模式则更有利于 PAM 验证功能的应用等。

2. xinetd 模式和 standalone 模式的区别

以 xinetd 模式运行的服务表示该服务进程并不以守护进程执行，以 ftpd 进程为例，若以 xinetd 模式运行这个服务，情况是这样的，虽然 FTP 服务本身会监听 21 号端口，但是以这种模式运行这个服务的话，21 号端口则会由 xinetd 进程来监听（此时 FTPD 服务并没有运行），如果你的网卡接收到有 21 号端口请求，则 xinetd 进程会去调用 FTPD 程序，将在 21 号端口接收到的请求数据移交给 ftpd 进程去处理，处理完后 ftpd 进程退出，而 xinetd 进程继续监听 21 号端口。这有点类似 Windows 的 svchost 进程。

而以 standalone 模式运行的服务则是服务进程，如 ftpd 以守护进程在内存中运行，接收到 21 号端口的请求后由 ftpd 进程 fork 出一个子进程来处理，而原进程则继续监听 21 号端口。

2.8.3 vsftpd 的数据连接模式

FTP 会话时包含了两个通道：一个叫控制通道，一个叫数据通道。

控制通道：控制通道是和 FTP 服务器进行沟通的通道，连接 FTP、发送 FTP 指令都是通过控制通道来完成的，它占用的端口是 21（服务器端）。

数据通道：数据通道是和 FTP 服务器进行文件传输或列表的通道，它占用的端口是 20（服务器端）。

在 FTP 协议中，控制连接均由客户端发起，而数据连接则有两种工作方式：PORT（主动）方式和 PASV（被动）方式。

1. PORT 模式（一般称之为主动方式）

FTP 客户端首先和 FTP Server 的 TCP 21 端口建立连接，通过这个通道发送命令，客户端需要在接收数据的时候在这个通道上发送 PORT 命令。PORT 命令包含了客户端用什么端口（一个大于 1024 的端口）接收数据。在传送数据的时候，服务器端通过自己的 TCP 20 端口发送数据。FTP Server 必须和客户端建立一个新的连接来传送数据。

2. PASV 模式（一般称之为被动方式）

PASV 模式在建立控制通道的时候和 PORT 模式类似，当客户端通过这个通道发送 PASV 命令的时候，FTP Server 打开一个位于 1024 和 5000 之间的随机端口，并且通知客户端在这个端口上传送数据的请求，然后 FTP Server 将通过这个端口进行数据传送，这个时候 FTP Server 不再需要建立一个新的和客户端之间的连接来传送数据了。

强烈建议大家用 Wireshark 或 sniffer 或科来等工具进行抓包实验，以便分清这两种模式下的数据包流向，这样就可以更清楚这两种模式的运作方式了。

21 端口在这两种模式下都是必须用到的。

注意 是否调用被动模式由客户端的 FTP 来决定，Windows XP 下自带的工具行 ftp 默认是主动模式传输数据。

2.8.4 vsftpd 到底安全在哪里

vsftpd 为什么安全呢？到底安全在哪里？我们看看下面的内容：

为了建构一个以安全为主的 FTP 服务器，vsftpd 针对操作系统的“程序的权限”概念来设计，如果你读过基础篇的程序与资源管理章节的话，你应该会晓得系统上面所执行的程序都会引发一个程序，我们称为 PID（Process ID），这个 PID 在系统上面能进行的任务与它拥有的权限有关。也就是说，PID 拥有的权限等级越高，它能够进行的任务就越多。举例来说，使用 root 身份所触发的 PID 通常拥有可以进行任何工作的权限等级。

不过，万一触发这个 PID 的程序（program）有漏洞而导致被黑客所攻击而取得此 PID 使用权时，那么黑客将会取得这个 PID 拥有的权限！所以，近来发展的套件都会尽量的将服务取得的 PID 权限降低，使得该服务即使不小心被入侵了，入侵者也无法得到有效的系统管理权限，这样会让我们的系统较为安全的，vsftpd 就是基于这种想法而设计的。

除了 PID 方面的权限之外，vsftpd 也支持 chroot 这个函式的功能，chroot 顾名思义就是 change root directory 的意思，那个 root 指的是“根目录”而非系统管理员。它可以将某个特定的目录变成根目录，所以与该目录没有关系的其他目录就不会被误用了。

举例来说，如果你以匿名身份登录我们的 FTP 服务的话，通常你会被限定在 /var/ftp 目录下工作，而你看到的根目录其实就只是 /var/ftp，至于如 /etc、/home、/usr 等其他目录你就看不到了！这样一来即使这个 FTP 服务被攻击了，没有关系，入侵者还是仅能在 /var/ftp 里面跑来跑去而已，而无法使用 Linux/Unix 的完整功能。自然我们的系统也就会比较安全了。

vsftpd 是基于上面的说明来设计的一个较为安全的 FTP 服务器软件，它具有如下的特点：

□ vsftpd 这个服务的启动者身份为一般使用者，所以对于 Linux 系统的使用权限较低，对于

Linux 系统的危害就相对的减低了。此外，vsftpd 也利用 chroot() 这个函式进行改换根目录的动作，使得系统工具不会被 vsftpd 这个服务所误用。

- 任何需要具有较高执行权限的 vsftpd 指令均以一个特殊的上层程序（parent process）所控制，该上层程序享有的较高执行权限功能已经被限制得相当低了，并以不影响 Linux 本身的系统为准。
- 绝大部分 FTP 会使用到的额外指令功能（dir、ls、cd 等）都已经被整合到 vsftpd 主程序当中了，因此理论上 vsftpd 不需要使用到额外的系统提供的指令，所以在 chroot 的情况下，vsftpd 不但可以顺利运作，且不需要额外功能，对于系统来说也比较安全。
- 来自客户端并且想要使用这个上层程序所提供的较高执行权限的 vsftpd 指令的所有需求，均被视为“不可信任的要求”，必须要经过相当程度的身份确认后，才能使用该上层程序的功能，例如 chown()、Login 的要求等动作。
- 此外，上面提到的上层程序中，依然使用 chroot() 的功能来限制使用者的执行权限。

基于以上特点，vsftpd 是比较安全的 FTP，这也是几个流行 Linux 发行版本默认安装的 FTP。如果大家考虑架设公网上的 FTP 的话，可以考虑用 vsftpd 来架设。

2.8.5 在 FreeBSD8.1 下配置 vsftpd 服务器

如何在 FreeBSD8.1 下配置 vsftpd 服务器呢？详细过程如下所示。

(1) 利用 ports 安装 vsftpd，如下所示：

```
cd/usr/ports/ftp/vsftpd
make install clean
```

记得把 make config 里面的两个选项都启动（前两项）。

(2) 配置 vsftpd 的基础配置。在 /etc/rc.conf 中添加以下内容，让 vsftpd 开机即可启动：

```
vsftpd_enable="YES"
```

在 /usr/local/etc/vsftpd.conf 中添加如下内容：

```
listen=YES
```

这表示将 vsftpd 以 standalone 的方式启动。

(3) 修改 /usr/local/etc/vsftpd.conf，配置文件如下：

```
anonymous_enable=NO
```

不允许匿名用户登录，主要是基于安全因素的考虑。

```
local_enable=YES
```

允许本地用户以 FTP 用户登录，将其作为内网使用，方便是我们要考虑的第一因素，所以在这里就没有配置虚拟用户了。

```
write_enable=YES
```

表示允许本地用户具有写权限。

```
local_umask=022
```


设置创建文件权限的反掩码，此处为 022，则新建文件的权限为 $666 - 022 = 644$ (rw-r--r--); 新建目录的权限为 $777 - 022 = 755$ (rwxr-xr-x)。这种方法其实也有误差，并不能百分百保证它的正确性，有兴趣的朋友可深入研究一下。

```
chroot_local_user=YES
```

表示限制本地用户的目录，不允许随意切换。

(4) 启动 vsftpd 服务，命令如下：

```
/usr/local/etc/rc.d/vsftpd start
```

下面进行登录测试。在这里我们可以用 FTP 尝试登录一下，你会发现匿名用户 FTP 和 anonymous 用户均不能登录。本地用户可以登录，在 Windows XP 下的工具我推荐 FileZilla Client 开源的东西就是好用，大家可以在天空软件园 <http://www.skycn.com> 下载尝试，工作界面如图 2-52 所示。

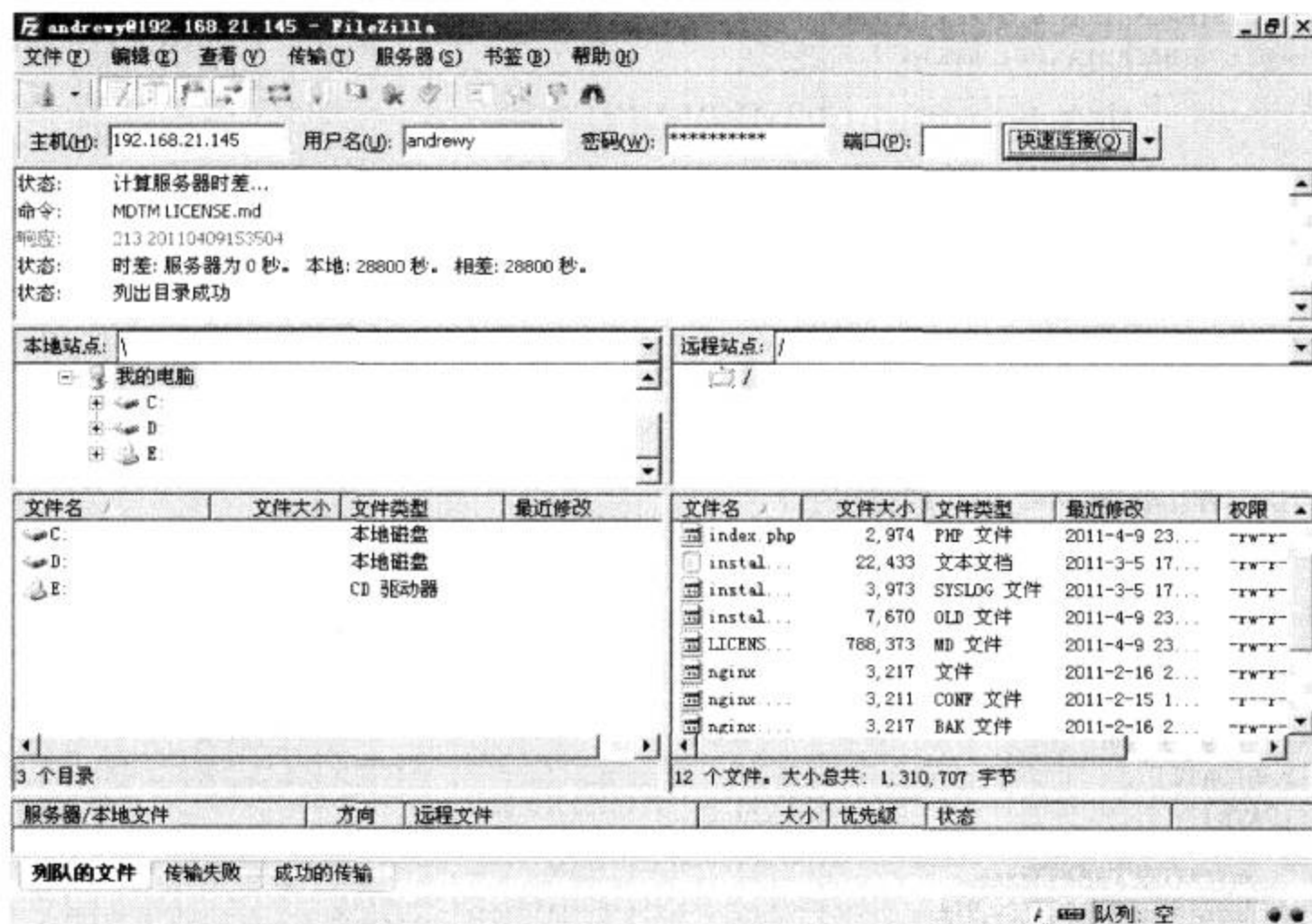


图 2-52 FileZilla Client 工作界面图示

有兴趣的朋友还可以尝试 Xmanager3.0 自带的 Xftp，它本身也支持 FTP 下载，嫌麻烦的朋友可直接用 Xftp，效果也不错的。Xftp 自身也可以用于 Windows 机器与 Linux/Unix 机器之间进行资料的上传下载，无需 FTP 服务开启，它直接通过 22 端口传输（类似于 WinSCP）。

2.8.6 用 vsftpd 作 Linux/Unix 之间的异地备份

本脚本的设计思想为：对 /home/andrewyu/ 下的重要目录 research 资料进行备份，首先在本地备份，每 10 天为一个轮询，然后以 backup 用户上传至 FTP 服务器 192.168.4.45 上。由于资料比较

小，我在 FTP 上面就没有做轮询处理。此脚本目前已正常运行一年多了，有兴趣的朋友可直接拿来使用。

此脚本在 FreeBSD8.0 x86_64 | FreeBSD8.1 x86_64 生产服务器下通过，代码如下：

```
#!/bin/bash
# Created by Andrew.Yu.
# 2010-04-23
SVNDIR=/home/andrewy/research
DATE='date +%Y-%m-%d'
OLDDATE='date -v -10d +%Y-%m-%d'

BACKDIR=/data/backup/research
FILENAME=cvsbackup_'date +%Y-%m-%d'

if[ ! -d ${BACKDIR}/${DATE} ]; then
    mkdir ${BACKDIR}/${DATE}
fi

if[ -d ${BACKDIR}/${OLDDATE} ]; then
    rm -rf ${BACKDIR}/${OLDDATE}
fi

HOST=192.168.4.45
FTP_USERNAME=backup
FTP_PASSWORD=backup

cd ${BACKDIR}/${DATE}
tar zcvf $FILENAME.tar.gz $SVNDIR

ftp -i -n -v << !
open ${HOST}
user ${FTP_USERNAME} ${FTP_PASSWORD}
bin
rm -rf ${OLDDATE}
mkdir ${DATE}
cd ${DATE}
mput*
bye
!
```

备份服务器是一台 Centos5.5 x86_64 的机器，配置了 vsftpd 服务，我们可以通过组合使用如下命令直接得出 vsftpd.conf 中有效的文件内容，如下所示：

```
grep -v "^#" /etc/vsftpd/vsftpd.conf | grep -v '^$'
local_enable=YES
write_enable=YES
local_umask=022
dirmessage_enable=YES
xferlog_enable=YES
connect_from_port_20=YES
xferlog_std_format=YES
```

```
listen=YES
chroot_local_user=YES
pam_service_name=vsftpd
userlist_enable=YES
tcp_wrappers=YES
```

chroot_local_user = YES 这句话的作用我要重点强调一下。它的作用是限制所有本地用户在登录 vsftpd 服务器时，只能在自己的 home 目录，基于安全的考虑，在编写脚本的过程中也考虑到了这点。关于 SHELL 备份脚本我在第 5 章会重点讲述。

vsftpd 服务器的特点不仅是安全，它的稳定性也是有目共睹的。我经常遇见放在公网上面的 vsftpd 的 FTP 服务器，几年没有重启和发生 crash 情况（最久的一个是 7 年没有重启了）。大家如果有架设 FTP 服务器需求的话，不妨考虑用 vsftpd 来作 FTP 服务器。如果对权限细化方面更有需求，也可以使用 Pureftpd，这款 FTP 软件的功能也非常强大。

2.9 在 FreeBSD8.1 和 Centos5.5 下搭建 PHP 与 Java 应用环境

在工作中，经常会有要求搭建环境的需求。这里所说的环境，即程序的应用环境，现在的程序种类繁多，有 Java、PHP、.NET，还有 Perl、Python 等，程序写出来以后就要一个环境运行，如果能熟练地在系统下配置程序的应用环境，会给我们的工作带来许多帮助，以下内容尤其适合于开发人员掌握。

2.9.1 在 FreeBSD8.1 下搭建 FAMP 环境

众所周知，在生产环境下配置 Apache + PHP 5 环境或 Nginx + PHP 5 环境还是很复杂的，比如大家熟悉的 LAMP 或 NMAP 架构，就算按照详细的部署文档来安装的话也是一个漫长而麻烦的过程。但这一切如果在 FreeBSD8.1 下用 ports 就简化了，我感觉它特别适合开发环境。下面给出 FreeBSD8.1 下的部署过程，以便与大家交流共享。详细安装过程如下（为了简化操作，我这里直接采用 root 操作）。

（1）安装 Apache22，如下所示：

```
cd /usr/ports/www/apache22
make install clean
```

去掉 IPv6，添加 MySQL。

Apache22 在 FreeBSD8.1 下安装时启动会报错，以下方法可修复 Apache22 在 FreeBSD8.1 下启动时的的问题，如下所示：

```
cd /usr/src/sys/modules/accf_data
make
make install clean
cd /usr/src/sys/modules/accf_http
make
make install clean
```

用 vim/etc/loader.conf 文件，加入以下代码：

```
accf_data_load="YES"
```

```
accf_http_load="YES"
```

编辑 `etc/rc.conf` 添加配置信息并启动，即加入以下代码：

```
apache22_enable="YES"
apache22_http_accept_enable="YES"
```

这时候启动 Apache22 就应该没什么问题了。我们输入如下代码来启动 Apache：

```
/usr/local/etc/rc.d/apache22 start
```

(2) 修改 Apache22 的配置文件 `/usr/local/etc/apache22/httpd.conf`。

安装完成后，备份 `/usr/local/etc/apache22/httpd.conf` 文件，命令如下：

```
cp /usr/local/etc/apache22/httpd.conf /usr/local/etc/apache22/httpd.conf.bak
```

编辑 `/usr/local/etc/apache22/httpd.conf` 文件，以使 Apache22 Server 支持 PHP。

```
vim /usr/local/etc/apache22/httpd.conf
```

在 `AddType application/x-gzip.gz.tgz` 下添加以下内容：

```
AddType application/x-httpd-php .php
AddType application/x-httpd-php-source .phps
```

添加 `index.php` 到 `DirectoryIndex` 主目录索引。

```
DirectoryIndex index.php index.html
```

修改 Apache22 的默认配置为自己的需求配置，如下所示：

```
DocumentRoot "/home/www"
```

`DocumentRoot` 指定了存放 Web 的路径，相当于根路径，根据自己的需求更改。

```
<Directory "/home/www">
```

(3) 安装 PHP5.2.17，如下所示：

```
cd /usr/ports/lang/php52
make install clean
```

选择如下安装选项：

- ☐ 去掉 IPv6。
- ☐ 添加 CLI、CGI、APACHE、SUHOSION、FASTCGI、PATHINFO。

(4) 安装 PHP5.2 扩展包，如下所示：

```
cd /usr/ports/lang/php52-extensions
make config
make install clean
```

在 `make config` 时添加如下选项：

BZ2、CALENDAR、CTYPE、GD、GETTEXT、ICONV、MBSTRING、MCRYPT、MHASH、MYSQL、MYSQLI、OPENSSL、PCRE、POSIX、SESSION、SOCKETS、TOKENIZER、ZIP、ZLIB。

以上仅仅是我的开发环境所需要的软件包，大家可根据自己的需求来选择。如果遗漏了也不要

紧，我们可以用 `phpize` 工具来动态加载。

我在线上服务器的 Centos5.5 下安装 LNMP 环境，一个个手动源码编译 PHP 5 的扩展包，感觉是件非常痛苦的事情。而在 FreeBSD8 下一切就都简单了，不过这个过程有点长，请大家合理化分配时间。

复制 `/usr/local/etc/php.ini-dist` 为 `/usr/local/etc/php.ini`，命令如下：

```
cp /usr/local/etc/php.ini-dist /usr/local/etc/php.ini
```

(5) 安装 Zend Optimizer 来优化 PHP 执行速度，如下所示：

```
cd /usr/ports/devel/ZendOptimizer
make install clean
```

安装完成提示如下：

```
***** You
have installed the ZendOptimizer package.
Edit /usr/local/etc/php.ini and add:
[ zend ]
zend_optimizer.optimization_level=15
zend_extension_manager.optimizer="/usr/local/lib/php/20060613/Optimizer"
zend_extension_manager.optimizer_ts="/usr/local/lib/php/20060613/Optimizer_TS"
zend_extension="/usr/local/lib/php/20060613/ZendExtensionManager.so"
zend_extension_ts="/usr/local/lib/php/20060613/ZendExtensionManager_TS.so"
*****
```

编辑 `php.ini`，复制以上 `[Zend]` 的内容到文本末尾。

(6) 重启 Apache Server，编写 `index.php` 测试一下看是否能支持 PHP。

在 `/home/www` 里面建立一个 `index.php`，输入以下内容：

```
<?
Phpinfo();
?>
```

我这里安装的版本为 5.2.17，新的 PHP 已更新至 5.3.3，但是老版本的 Zend Optimizer 对其不支持，所以本着稳定就好的原则，继续使用 5.2.17，大家可根据实际需求来选择。

(7) 安装 MySQL 5.5.14，如下所示（MySQL 的版本选择也可以通过 ports 来进行）：

```
cd /usr/ports/databases/mysql55-server
make install clean
```

安装完成后，需要编辑 `/etc/rc.conf` 文件来配置 MySQL，添加如下内容到 `/etc/rc.conf` 中：

```
mysql_enable="YES"
```

复制文件，如下所示：

```
cp /usr/local/share/mysql/my-medium.cnf/etc/my.conf
```

启动 MySQL 服务，命令如下：

```
/usr/local/etc/rc.d/mysql-server start
```

新装的 MySQL 是没有密码的，使用 `mysqladmin` 命令更改 MySQL 密码，命令如下：

```
/usr/local/bin/mysqladmin -uroot password '你的密码'
```

例如：

```
/usr/local/bin/mysqladmin -uroot password 12345678
```

(8) 安装图形化操作 MySQL 的工具 phpMyadmin。

1) 下载及安装过程，命令如下：

```
cd /usr/ports/databases/phpmyadmin
make fetch
```

这行命令的作用只是下载，不进行安装，因为 phpmyadmin 解压以后就能直接使用了，命令如下：

```
cp /usr/ports/distfiles/phpMyAdmin-3.3.2-all-languages.tar.bz2 /home/www
```

`/home/www` 为 Apache 的指定路径。

```
tar zxvf phpMyAdmin-3.3.2-all-languages.tar.bz2
mv phpMyAdmin-3.3.2-all-languages phpmyadmin
```

phpMyadmin 的文件名就是外部地址路径：`http://xxx.xxx.xxx.xxx/phpmyadmin`。

2) 配置 phpmyadmin 的过程如下所示：

```
cd /home/www/phpmyadmin
cp config.sample.inc.php config.inc.php
```

我们编辑一下 phpmyadmin 的配置文件 `config.inc.php`，如下：

```
vim config.inc.php
cfg['blowfish_secret'] = 'host';
```

上面的 `host` 可随便输入，不要留空，一定要设置。

```
cfg['Servers'][$i]['auth_type'] = 'cookie'
```

上面是设置认证方式，默认即可。

```
chmod 755 config.inc.php
```

上面是 `config.inc.php` 配置权限。

这时候你就可以在内网用 `root` 等账户进行登录管理了。phpMyadmin 的强大是有目共睹的，这也是它现在作为许多 Linux/FreeBSD 默认软件的原因之一。

值得注意的是，基于线上环境的严谨性，我一般采用 64 位的 Centos 系统，软件均采用源码安装。而以上所述，均是出于对开发环境的考虑，即快速方便地部署测试服务器，如果要用于线上环境，需要注意的细节还有许多，比如 PHP 要禁用危险的函数、Apache 要考虑其 Web 安全及 SSL 证书（一个不小心就要考虑支持多域名的 SSL 证书）等、MySQL 要考虑生产环境下的压力及备份等，这里就不细述了。在下一节里我向大家介绍一下 Centos5.5 下配置 LNMP 环境的详细过程。

2.9.2 在生产环境下配置 LNMP 环境^①

以下内容取自于我的项目实施文档，版本我采用的是当时最稳定的版本，大家可以按照我的文

^① 本节部分内容参考了张宴先生的博客 <http://blog.s135.com/nginx-php-v6/>，在此深表感谢。

档实施，也可以下载新源码包安装。服务器系统为 RHEL5.2 x86_64，安装的主要源码包为 Nginx0.8.15 + PHP5.2.6 + MySQL5.1.38，部分内容进行了精简，比如在原文档中 PHP 源码编译时，我是让它既支持 MySQL 又支持 MSSQL 的，而这里我只是让其支持了 MySQL。后来其他机房的 Windows Server2003 的 PHP 环境需要迁移至 LNMP 环境上，系统为清一色的 Centos5.5 x86_64，我正好也将此文档内容也升级为 Nginx0.8.46 + PHP5.2.14 + MySQL5.5.3 了。这样既可以用于自己的测试环境。也可将其用于生产环境，以下方法也适用于 32 位或 64 位 Centos5.x 版本。

Web 服务器均采用 HPDL380G6P，CPU 为英特尔至强 E5540@2.53GHz，双四核（Gainestown），内存为金士顿 DDR2 * 4 8G，硬盘为 RAID1 300GB，在其他机器（如 DELL 2950 和 DELL R710）上部署以下过程也很顺利。

1. 安装前的准备工作

(1) 保证 ntp 时间同步，命令如下：

```
yum -y install vixie-cron
```

我们可以每天 ntp 对时一次，即编辑/etc/crontab 文件，添加代码如下：

```
00 00*** root /usr/sbin/ntpdate ntp.api.bz > >/dev/null 2>&1
```

让 Centos5.5 的 crond 开机即启动，命令如下：

```
service crond start | chkconfig crond start
```

(2) 由于在安装前还需要安装软件库，我在内网配置了一台 yum 服务器（过程略），嫌麻烦的朋友可直接采用 Centos5.5 的源来安装以下软件库，命令如下：

```
yum -y install gcc gcc-c++ autoconf libjpeg libjpeg-devel libpng libpng-devel freetype freetype-devel libxml2 libxml2-devel zlib zlib-devel glibc glibc-devel glib2 glib2-devel bzip2 bzip2-devel ncurses ncurses-devel curl curl-devel e2fsprogs e2fsprogs-devel krb5 krb5-devel libidn libidn-devel openssl openssl-devel openldap openldap-devel nss_ldap openldap-clients openldap-servers
```

(3) 下载 LNMP 环境所需要的源码包，如下所示：

```
mkdir -p /usr/local/src
cd /usr/local/src
```

下载架构环境所需的源码包，其文件列表清单 list.txt 文件如下：

```
http://blog.s135.com/soft/linux/nginx_php/nginx/nginx-0.8.46.tar.gz
http://blog.s135.com/soft/linux/nginx_php/php/php-5.2.14.tar.gz
http://blog.s135.com/soft/linux/nginx_php/phpfpm/php-5.2.14-fpm-0.5.14.diff.gz
http://blog.s135.com/soft/linux/nginx_php/mysql/mysql-5.5.3-m3.tar.gz
http://blog.s135.com/soft/linux/nginx_php/libiconv/libiconv-1.13.1.tar.gz
http://blog.s135.com/soft/linux/nginx_php/mcrypt/libmcrypt-2.5.8.tar.gz
http://blog.s135.com/soft/linux/nginx_php/mcrypt/mcrypt-2.6.8.tar.gz
http://blog.s135.com/soft/linux/nginx_php/memcache/memcache-2.2.5.tgz
http://blog.s135.com/soft/linux/nginx_php/mhash/mhash-0.9.9.9.tar.gz
```

http://blog.s135.com/soft/linux/nginx_php/pcre/pcre-8.10.tar.gz
http://blog.s135.com/soft/linux/nginx_php/eaccelerator/eaccelerator-0.9.6.1.tar.bz2
http://blog.s135.com/soft/linux/nginx_php/pdo/PDO_MYSQL-1.0.2.tgz
http://blog.s135.com/soft/linux/nginx_php/imagick/ImageMagick.tar.gz
http://blog.s135.com/soft/linux/nginx_php/imagick/imagick-2.3.0.tgz

开始下载，可以用如下命令来实现：

```
wget -i /usr/local/src/list.txt
```

2. 安装 php-5.2.14 所需要的支持库

以下过程我们可以写一个 SHELL 脚本，让它全自动完成，免得一个个地手动输入命令，SHELL 脚本如下：

```
#!/bin/bash
tar zxvf libiconv-1.13.1.tar.gz
cd libiconv-1.13.1/
./configure --prefix=/usr/local
make
make install
cd ../

tar zxvf libmcrypt-2.5.8.tar.gz
cd libmcrypt-2.5.8/
./configure
make
make install
/sbin/ldconfig
cd libltdl/
./configure --enable-ltdl-install
make
make install
cd ../../

tar zxvf mhash-0.9.9.9.tar.gz
cd mhash-0.9.9.9/
./configure
make
make install
cd ../

ln -s /usr/local/lib/libmcrypt.la /usr/lib/libmcrypt.la
ln -s /usr/local/lib/libmcrypt.so /usr/lib/libmcrypt.so
ln -s /usr/local/lib/libmcrypt.so.4 /usr/lib/libmcrypt.so.4
ln -s /usr/local/lib/libmcrypt.so.4.4.8 /usr/lib/libmcrypt.so.4.4.8
ln -s /usr/local/lib/libmhash.a /usr/lib/libmhash.a
ln -s /usr/local/lib/libmhash.la /usr/lib/libmhash.la
ln -s /usr/local/lib/libmhash.so /usr/lib/libmhash.so
ln -s /usr/local/lib/libmhash.so.2 /usr/lib/libmhash.so.2
ln -s /usr/local/lib/libmhash.so.2.0.1 /usr/lib/libmhash.so.2.0.1
```



```
ln -s /usr/local/bin/libmccrypt-config /usr/bin/libmccrypt-config

tar zxvf mccrypt-2.6.8.tar.gz
cd mccrypt-2.6.8/
/sbin/ldconfig
./configure
make
make install
cd ../
```

3. 源码编译安装 MySQL 5.5.3-m3

(1) 建立运行 MySQL 数据库的用户组及用户 mysql，完成下面的安装过程，命令如下：

```
groupadd mysql
useradd -g mysql mysql
tar zxvf mysql-5.5.3-m3.tar.gz
cd mysql-5.5.3-m3/
```

MySQL 5.5.3-m3 的编译参数如下：

```
./configure --prefix=/usr/local/webserver/mysql --enable-assembler --with-extra-charsets=complex --enable-thread-safe-client --with-big-tables --with-readline --with-ssl --with-embedded-server --enable-local-infile --with-plugins=partition,innobase,myisam,myisamchk
make && make install
```

给 mysql 运行所在的目录以执行权限，将其所有权给予 mysql: mysql，命令如下：

```
chmod +w /usr/local/webserver/mysql
chown -R mysql:mysql /usr/local/webserver/mysql
cd ../
```

(2) 创建 MySQL 数据库存放的目录，这里建立 reallog 目录用于存放 mysql-relay-bin.xxx 文件，它可以存放 slave 端的 I/O 线程从 master 端所读取的 binary log 信息，然后由 slave 端的 SQL 线程从该 relay log 中读取并解析相应的日志信息，转化成 master 所执行的 query 语句，接着在 slave 端应用，/data/mysql/3306/data 用于存放 MySQL 数据，/data/mysql/3306/binlog 用于此 MySQL 运行时所产生的二进制文件，如下所示：

```
mkdir -p /data/mysql/3306/data/
mkdir -p /data/mysql/3306/binlog/
mkdir -p /data/mysql/3306/relaylog/
chown -R mysql:mysql /data/mysql/
```

(3) 以 mysql 用户账号的身份建立数据表，命令如下：

```
/usr/local/webserver/mysql/bin/mysql_install_db
--basedir=/usr/local/webserver/mysql
--datadir=/data/mysql/3306/data --user=mysql
```

成功后应该有如下显示：

```
/usr/local/webserver/mysql/bin/mysqladmin -u root password 'new-password'
/usr/local/webserver/mysql/bin/mysqladmin -u root -h server.cn7788.com password 'new-password'
```

Alternatively you can run:

```
/usr/local/webserver/mysql/bin/mysql_secure_installation
which will also give you the option of removing the test
databases and anonymous user created by default. This is
strongly recommended for production servers.
```

See the manual for more instructions.

You can start the MySQL daemon with:

```
cd/usr/local/webserver/mysql ;/usr/local/webserver/mysql/bin/mysqld_safe &
```

You can test the MySQL daemon with `mysql - test - run.pl`

```
cd/usr/local/webserver/mysql/mysql - test ; perl mysql - test - run.pl
```

Please report any problems with the `/usr/local/webserver/mysql/scripts/mysqlbug` script!

(4) 创建 `my.cnf` 配置文件，我将其放在 `/data/mysql/3306` 下，即 `/data/mysql/3306` 文件。文件内容如下所示：

```
[client]
character-set-server = utf8
port = 3306
socket = /tmp/mysql.sock

[mysqld]
character-set-server = utf8
replicate-ignore-db = mysql
replicate-ignore-db = test
replicate-ignore-db = information_schema
user = mysql
port = 3306
socket = /tmp/mysql.sock
basedir = /usr/local/webserver/mysql
datadir = /data/mysql/3306/data
log-error = /data/mysql/3306/mysql_error.log
pid-file = /data/mysql/3306/mysql.pid
open_files_limit = 10240
back_log = 600
max_connections = 5000
max_connect_errors = 6000
table_cache = 614
external-locking = FALSE
max_allowed_packet = 32M
sort_buffer_size = 1M
join_buffer_size = 1M
thread_cache_size = 300
#thread_concurrency = 8
query_cache_size = 512M
query_cache_limit = 2M
query_cache_min_res_unit = 2k
default-storage-engine = MyISAM
thread_stack = 192K
transaction_isolation = READ-COMMITTED
tmp_table_size = 246M
max_heap_table_size = 246M
```

```

long_query_time = 3
log-slave-updates
log-bin = /data/mysql/3306/binlog/binlog
binlog_cache_size = 4M
binlog_format = MIXED
max_binlog_cache_size = 8M
max_binlog_size = 1G
relay-log-index = /data/mysql/3306/relaylog/relaylog
relay-log-info-file = /data/mysql/3306/relaylog/relaylog
relay-log = /data/mysql/3306/relaylog/relaylog
expire_logs_days = 30
key_buffer_size = 256M
read_buffer_size = 1M
read_rnd_buffer_size = 16M
bulk_insert_buffer_size = 64M
myisam_sort_buffer_size = 128M
myisam_max_sort_file_size = 10G
myisam_repair_threads = 1
myisam_recover

interactive_timeout = 120
wait_timeout = 120

skip-name-resolve
slave-skip-errors = 1032,1062,126,1114,1146,1048,1396

server-id = 1

innodb_additional_mem_pool_size = 16M
innodb_buffer_pool_size = 512M
innodb_data_file_path = ibdata1:256M:autoextend
innodb_file_io_threads = 4
innodb_thread_concurrency = 8
innodb_flush_log_at_trx_commit = 2
innodb_log_buffer_size = 16M
innodb_log_file_size = 128M
innodb_log_files_in_group = 3
innodb_max_dirty_pages_pct = 90
innodb_lock_wait_timeout = 120
innodb_file_per_table = 0

log-slow-queries = /data/mysql/3306/slow.log
long_query_time = 10

[mysqldump]
quick
max_allowed_packet = 32M

```

(5) 用命令启动 MySQL 数据库，无论是内部开发环境还是生产环境，我们的 MySQL 数据库启动后就不会关闭了，没有特殊情况也不会重启。所以也可以将以下启动 MySQL 的命令置于/etc/

rc.local 文件内, 命令如下:

```
/usr/local/webserver/mysql/bin/mysqld_safe
--defaults-file=/data/mysql/3306/my.cnf
```

这样我们只要重启服务器, 就可以启动 MySQL 数据库了。我们还可以用命令验证, 命令如下:

```
lsof -i:3306
COMMAND PID USER FD TYPE DEVICE SIZE NODE NAME
mysqld 18286 mysql 14u IPv6 92763 TCP* :mysql (LISTEN)
```

4. 编译安装 php-5.2.14

我们进入 /usr/local/src 目录后, 就可以编译安装 php-5.2.14, 如下所示:

```
tar zxvf php-5.2.14.tar.gz
gzip -cd php-5.2.14-fpm-0.5.14.diff.gz | patch -d php-5.2.14 -p1
```

不打这个补丁的话, 会产生无 sbin 目录的错误。

```
cd php-5.2.14/
```

php-5.2.14 的编译参数如下:

```
./configure --prefix=/usr/local/webserver/php
--with-config-file-path=/usr/local/webserver/php/etc
--with-mysql=/usr/local/webserver/mysql
--with-mysqli=/usr/local/webserver/mysql/bin/mysql_config
--with-iconv-dir=/usr/local --with-freetype-dir
--with-jpeg-dir --with-png-dir --with-zlib --with-libxml-dir=/usr --enable-xml
--disable-rpath --enable-discard-path --enable-safe-mode
--enable-bcmath --enable-shmop --enable-sysvsem
--enable-inline-optimization --with-curl --with-curlwrappers
--enable-mbregex --enable-fastcgi --enable-fpm
--enable-force-cgi-redirect --enable-mbstring --with-mcrypt
--with-gd --enable-gd-native-ttf --with-openssl --with-mhash
--enable-pcntl --enable-sockets --with-ldap --with-ldap-sasl
--with-xmlrpc --enable-zip --enable-soap
make ZEND_EXTRA_LIBS='-liconv'
```

为免得发生 libiconv 报错, 所以带上 ZEND 参数, 这个问题要是经常编译 PHP 的朋友非常熟悉的。

```
make install
```

源码编译安装完成后, 我们为 php 创建 php.ini 文件, 如下所示:

```
cp php.ini-dist /usr/local/webserver/php/etc/php.ini
cd ../
```

5. 安装 php5 扩展模块

以下过程也可以写成脚本形式, 免得手动一个个输入, 内容如下:

```
#!/bin/bash
tar zxvf memcache-2.2.5.tgz
```



```

cd memcache-2.2.5/
/usr/local/webserver/php/bin/phpize
./configure --with-php-config=/usr/local/webserver/php/bin/php-config
make
make install
cd ../

tar jxvf eaccelerator-0.9.6.1.tar.bz2
cd eaccelerator-0.9.6.1/
/usr/local/webserver/php/bin/phpize
./configure --enable-eaccelerator=shared --with-php-config=/usr/local/webserver/php/
bin/php-config
make
make install
cd ../

tar zxvf PDO_MYSQL-1.0.2.tgz
cd PDO_MYSQL-1.0.2/
/usr/local/webserver/php/bin/phpize
./configure --with-php-config=/usr/local/webserver/php/bin/php-config --with-pdo-mysql
=/usr/local/webserver/mysql
make
make install
cd ../

tar zxvf ImageMagick.tar.gz
cd ImageMagick-6.5.1-2/
./configure
make
make install
cd ../

tar zxvf imagick-2.3.0.tgz
cd imagick-2.3.0/
/usr/local/webserver/php/bin/phpize
./configure --with-php-config=/usr/local/webserver/php/bin/php-config
make
make install
cd ../

```

修改 php.ini 文件，加载动态模块，让 PHP5.2.14 能顺利启动。

手动修改，查找：

/usr/local/webserver/php/etc/php.ini 中的 extension_dir = "./"

将其修改如下：

```

extension_dir =
"/usr/local/webserver/php/lib/php/extensions/no-debug-non-zts-20060613/"

```

并在此行后增加以下几行内容，然后保存：

```

cd memcache-2.2.5/
/usr/local/webserver/php/bin/phpize
./configure --with-php-config=/usr/local/webserver/php/bin/php-config
make
make install
cd ../

tar jxvf eaccelerator-0.9.6.1.tar.bz2
cd eaccelerator-0.9.6.1/
/usr/local/webserver/php/bin/phpize
./configure --enable-eaccelerator=shared --with-php-config=/usr/local/webserver/php/
bin/php-config
make
make install
cd ../

tar zxvf PDO_MYSQL-1.0.2.tgz
cd PDO_MYSQL-1.0.2/
/usr/local/webserver/php/bin/phpize
./configure --with-php-config=/usr/local/webserver/php/bin/php-config --with-pdo-mysql
=/usr/local/webserver/mysql
make
make install
cd ../

tar zxvf ImageMagick.tar.gz
cd ImageMagick-6.5.1-2/
./configure
make
make install
cd ../

tar zxvf imagick-2.3.0.tgz
cd imagick-2.3.0/
/usr/local/webserver/php/bin/phpize
./configure --with-php-config=/usr/local/webserver/php/bin/php-config
make
make install
cd ../

```

修改 php.ini 文件，加载动态模块，让 PHP5.2.14 能顺利启动。

手动修改，查找：

/usr/local/webserver/php/etc/php.ini 中的 extension_dir = "./"

将其修改如下：

```

extension_dir =
"/usr/local/webserver/php/lib/php/extensions/no-debug-non-zts-20060613/"

```

并在此行后增加以下几行内容，然后保存：

创建根路径，如下所示：

```
mkdir -p /data/logs
```

创建 Nginx 日志路径。分别为我的 Nginx 创建虚拟用户目录，如下所示：

```
cd /data/htdocs/www
mkdir adongweb baobei ImageVue very365
```

分别将其目录权限给予 www：www，命令如下：

```
chown -R www:www/data/logs
chown -R www:www/data/htdocs/www
```

(2) 创建 php-fpm 配置文件（php-fpm 是为 PHP 打的一个 FastCGI 管理补丁，可以平滑变更 php.ini 配置而无需重启 php-cgi）。在 /usr/local/webserver/php/etc/ 目录中创建 php-fpm.conf 文件，命令如下：

```
rm -f /usr/local/webserver/php/etc/php - fpm.conf
```

我们用 rm 命令删除原有的 php-fpm.conf 文件后，创建一个新的 /usr/local/webserver/php/etc/ php-fpm.conf 文件，文件内容如下：

```
<?xml version="1.0"?>
<configuration>

    All relative paths in this config are relative to php's install prefix

    <section name="global_options">

        Pid file
        <value name="pid_file">/usr/local/webserver/php/logs/php - fpm.pid</value>

        Error log file
        <value name="error_log">/usr/local/webserver/php/logs/php - fpm.log</value>

        Log level
        <value name="log_level">notice</value>

        When this amount of php processes exited with SIGSEGV or SIGBUS ...
        <value name="emergency_restart_threshold">10</value>

        ...in a less than this interval of time, a graceful restart will be initiated.
        Useful to work around accidental corruptions in accelerator's shared memory.
        <value name="emergency_restart_interval">1m</value>

        Time limit on waiting child's reaction on signals from master
        <value name="process_control_timeout">5s</value>

        Set to 'no' to debug fpm
        <value name="daemonize">yes</value>
```

```
</section>
```

```
<workers>
```

```
<section name = "pool">
```

Name of pool. Used in logs and stats.

```
<value name = "name">default</value>
```

Address to accept fastcgi requests on.

Valid syntax is 'ip.ad.re.ss:port' or just 'port' or '/path/to/unix/socket'

```
<value name = "listen_address">127.0.0.1:9000</value>
```

```
<value name = "listen_options">
```

Set listen(2) backlog

```
<value name = "backlog">-1</value>
```

Set permissions for unix socket, if one used.

In Linux read/write permissions must be set in order to allow connections from web server.

Many BSD-derived systems allow connections regardless of permissions.

```
<value name = "owner"></value>
```

```
<value name = "group"></value>
```

```
<value name = "mode">0666</value>
```

```
</value>
```

Additional php.ini defines, specific to this pool of workers.

```
<value name = "php_defines">
```

```
<value name = "sendmail_path">/usr/sbin/sendmail -t -i</value>
```

```
<value name = "display_errors">0</value>
```

```
</value>
```

Unix user of processes

```
<value name = "user">www</value>
```

Unix group of processes

```
<value name = "group">www</value>
```

Process manager settings

```
<value name = "pm">
```

Sets style of controlling worker process count.

Valid values are 'static' and 'apache-like'

```
<value name = "style">static</value>
```

Sets the limit on the number of simultaneous requests that will be served.

Equivalent to Apache MaxClients directive.

Equivalent to PHP_FCGI_CHILDREN environment in original php.fcgi

Used with any pm_style.


```
<value name = "max_children" >300 </value >
```

Settings group for 'apache-like' pm style

```
<value name = "apache_like" >
```

Sets the number of server processes created on startup.

Used only when 'apache-like' pm_style is selected

```
<value name = "StartServers" >20 </value >
```

Sets the desired minimum number of idle server processes.

Used only when 'apache-like' pm_style is selected

```
<value name = "MinSpareServers" >5 </value >
```

Sets the desired maximum number of idle server processes.

Used only when 'apache-like' pm_style is selected

```
<value name = "MaxSpareServers" >35 </value >
```

```
</value >
```

```
</value >
```

The timeout (in seconds) for serving a single request after which the worker process will be terminated

Should be used when 'max_execution_time' ini option does not stop script execution for some reason

'0s' means 'off'

```
<value name = "request_terminate_timeout" >0s </value >
```

The timeout (in seconds) for serving of single request after which a php backtrace will be dumped to slow.log file

'0s' means 'off'

```
<value name = "request_slowlog_timeout" >0s </value >
```

The log file for slow requests

```
<value name = "slowlog" >logs/slow.log </value >
```

Set open file desc rlimit

```
<value name = "rlimit_files" >65535 </value >
```

Set max core size rlimit

```
<value name = "rlimit_core" >0 </value >
```

Chroot to this directory at the start, absolute path

```
<value name = "chroot" > </value >
```

Chdir to this directory at the start, absolute path

```
<value name = "chdir" > </value >
```

Redirect workers' stdout and stderr into main error log.

If not set, they will be redirected to/dev/null, according to FastCGI specs

```
<value name="catch_workers_output">yes</value>
```

How much requests each process should execute before respawn.

Useful to work around memory leaks in 3rd party libraries.

For endless request processing please specify 0

Equivalent to PHP_FCGI_MAX_REQUESTS

```
<value name="max_requests">1024</value>
```

Comma separated list of ipv4 addresses of FastCGI clients that allowed to connect.

Equivalent to FCGI_WEB_SERVER_ADDRS environment in original php.fcgi (5.2.2+)

Makes sense only with AF_INET listening socket.

```
<value name="allowed_clients">127..0.0.1</value>
```

Pass environment variables like LD_LIBRARY_PATH

All \$VARIABLEs are taken from current environment

```
<value name="environment">
```

```
<value name="HOSTNAME">$HOSTNAME</value>
```

```
<value name="PATH">/usr/local/bin:/usr/bin:/bin</value>
```

```
<value name="TMP">/tmp</value>
```

```
<value name="TMPDIR">/tmp</value>
```

```
<value name="TEMP">/tmp</value>
```

```
<value name="OSTYPE">$OSTYPE</value>
```

```
<value name="MACHTYPE">$MACHTYPE</value>
```

```
<value name="MALLOC_CHECK_">2</value>
```

```
</value>
```

```
</section>
```

```
</workers>
```

```
</configuration>
```

(3) 启动 php-cgi 进程，监听 127.0.0.1 的 9000 端口，进程数为 300（生产环境下的服务器为 300 ~ 500 个均可），用户为 www，如下所示：

```
ulimit -SHn 65535
```

```
/usr/local/webserver/php/sbin/php - fpm start
```

注意 usr/local/webserver/php/sbin/php-fpm 还有其他参数，包括：start | stop | quit | restart | reload | logrotate。修改 php.ini 后不重启 php-cgi，重新加载配置文件使用 reload。

(4) 安装并配置 Nginx0.8.46。

首先安装 PCRE，让 Nginx0.8.46 能支持正则表达式，命令如下：

```
tar zxvf pcre-8.46.tar.gz
```

```
cd pcre-8.10/
```

```
./configure
```

```
make && make install
```

```
cd ../
```

再接着安装 Nginx0.8.46, 命令如下:

```
tar zxvf nginx-0.8.46.tar.gz
cd nginx-0.8.46/
./configure --user=www --group=www
--prefix=/usr/local/webserver/nginx --with-http_stub_status_module
--with-http_ssl_module
make && make install
cd ../
```

(5) 配置 Nginx 的 conf 文件, 我们可以用 Vim 修改 /usr/local/webserver/nginx/conf/nginx.conf 文件, 内容如下:

```
user www www;
worker_processes 8;
error_log /data/logs/nginx_error.log crit;
pid /usr/local/webserver/nginx/nginx.pid;
#Specifies the value for maximum file descriptors that can be opened by this process.
worker_rlimit_nofile 65535;

events
{
    use epoll;
    worker_connections 65535;
}

http
{
    include mime.types;
    default_type application/octet-stream;

    #charset gb2312;

    server_names_hash_bucket_size 128;
    client_header_buffer_size 32k;
    large_client_header_buffers 4 32k;
    client_max_body_size 8m;
    sendfile on;
    tcp_nopush on;

    keepalive_timeout 60;

    tcp_nodelay on;

    fastcgi_connect_timeout 300;
    fastcgi_send_timeout 300;
    fastcgi_read_timeout 300;
    fastcgi_buffer_size 64k;
    fastcgi_buffers 4 64k;
    fastcgi_busy_buffers_size 128k;
    fastcgi_temp_file_write_size 128k;
```

```

gzip on;
gzip_min_length 1k;
gzip_buffers 4 16k;
gzip_http_version 1.0;
gzip_comp_level 2;
gzip_types text/plain application/x-javascript text/css application/xml;
gzip_vary on;

#limit_zone crawler $binary_remote_addr 10m;

server
{
    listen 80 default;
    server_name _;
    index index.html index.htm index.php;
    root /data/htdocs/www;
    #server_name_in_redirect off;

    location ~ .*\. (php|php5)? $
    {
        #fastcgi_pass unix:/tmp/php-cgi.sock;
        fastcgi_pass 127.0.0.1:9000;
        fastcgi_index index.php;
        include fcgi.conf;
    }

    location ~ .*\. (gif|jpg|jpeg|png|bmp|swf)$
    {
        expires 30d;
    }

    location ~ .*\. (js|css)? $
    {
        expires 1h;
    }

}

server
{
    listen 80;
    server_name www.adongstudio.com;
    index index.html index.htm index.php;
    root /data/htdocs/www/adongweb;

    #limit_conn crawler 20;

    location ~ .*\. (php|php5)? $
    {

```



```

#fastcgi_pass unix:/tmp/php-cgi.sock;
fastcgi_pass 127.0.0.1:9000;
fastcgi_index index.php;
include fcgi.conf;
}

location ~.*\.(gif|jpg|jpeg|png|bmp|swf)$
{
    expires      30d;
}

location ~.*\.(js|css)? $
{
    expires      1h;
}

log_format access '$remote_addr - $remote_user[ $time_local] "$request" '
    '$status $body_bytes_sent "$http_referer" '
    '" $http_user_agent" $http_x_forwarded_for';
access_log /data/logs/access.log access;
}

server
{
    listen      80;
    server_name www.longfeistudio.com;
    index index.html index.htm index.php;
    root /data/htdocs/www/ImageVue;

    #limit_conn crawler 20;

    location ~.*\.(php|php5)? $
    {
        #fastcgi_pass unix:/tmp/php-cgi.sock;
        fastcgi_pass 127.0.0.1:9000;
        fastcgi_index index.php;
        include fcgi.conf;
    }

    location ~.*\.(gif|jpg|jpeg|png|bmp|swf) $
    {
        expires      30d;
    }

    location ~.*\.(js|css)? $
    {
        expires      1h;
    }

    access_log off;
}

```

```

    }
server
{
    listen      80;
    server_name www.hongyanbike.com;
    index index.html index.htm index.php;
    root /data/htdocs/www/xhui/hybike;

    location ~ .*\. (php | php5)? $
    {
        #fastcgi_pass unix:/tmp/php-cgi.sock;
        fastcgi_pass 127.0.0.1:9000;
        fastcgi_index index.php;
        include fcgi.conf;
    }

    location ~ .*\. (gif | jpg | jpeg | png | bmp | swf) $
    {
        expires      30d;
    }

    location ~ .*\. (js | css)? $
    {
        expires      1h;
    }

    access_log off;
}
server
{
    listen      80;
    server_name www.very365.com mm.very365.com very365.com;
    index index.html index.htm index.php;
    root /data/htdocs/www/very365;
    location/
    {
        rewrite ^/(.*)/product/([0-9]+)/ $/seoproduct\.php\? spell = $1&productid = $2;
        rewrite ^/brand/(.*)/page/([0-9]+)/ $/seobrand\.php\? spell = $1&page = $2;
        rewrite ^/brand/(.*)/ $/seobrand\.php\? spell = $1;

    }
    location ~ .*\. (php | php5)? $
    {
        #fastcgi_pass unix:/tmp/php-cgi.sock;
        fastcgi_pass 127.0.0.1:9000;
        fastcgi_index index.php;
        include fcgi.conf;
        fastcgi_param SCRIPT_FILENAME /data/htdocs/www/very365 $fastcgi_script_name;
        fastcgi_param SCRIPT_NAME /data/htdocs/www/very365 $fastcgi_script_name;
    }
}

```

```

location ~.*\.(gif|jpg|jpeg|png|bmp|swf)$
{
    expires      30d;
}

location ~.*\.(js|css) $
{
    expires      1h;
}
access_log off;
}
server
{
    listen      80;
    server_name www.wqueen.cn wqueen.cn;

    index index.html index.htm index.php;
    root /data/htdocs/www/wqueen/bbs;

    location ~.*\.(php|php5)? $
    {
        #fastcgi_pass unix:/tmp/php-cgi.sock;
        fastcgi_pass 127.0.0.1:9000;
        fastcgi_index index.php;
        include fcgi.conf;
    }

    location ~.*\.(gif|jpg|jpeg|png|bmp|swf) $
    {
        expires      30d;
    }

    location ~.*\.(js|css)? $
    {
        expires      1h;
    }

    access_log off;
}

server
{
    listen      80;
    server_name baobei.wqueen.cn;
    index index.html index.htm index.php;
    root /data/htdocs/www/baobei;

    location ~.*\.(php|php5)? $
    {
        #fastcgi_pass unix:/tmp/php-cgi.sock;

```

```

    fastcgi_pass 127.0.0.1:9000;
    fastcgi_index index.php;
    include fcgi.conf;
}

location ~.*\.(gif|jpg|jpeg|png|bmp|swf)$
{
    expires      30d;
}

location ~.*\.(js|css)?$
{
    expires      1h;
}

access_log off;
}
}

```

其实 Nginx 的虚拟目录跟 Apache 的配置过程差不多，每一个 `server` 即对应一个虚拟主机，`server_name` 是此虚拟主机的域名，`root` 是此虚拟主机的根目录。当然，我们预先就应该在 DNS 做好相应的解析对应关系。另外，由于此 LNMP 主要用于博客和论坛，后面的虚拟主机我就没有配置相应的日志了，有兴趣的朋友可以根据我的配置文件自行修改设置，这里就不细述了。

(6) 在 `/usr/local/webserver/nginx/conf/` 目录中创建 `fcgi.conf` 文件。`/usr/local/webserver/nginx/conf/fcgi.conf` 文件的内容如下：

```

fastcgi_param GATEWAY_INTERFACE CGI/1.1;
fastcgi_param SERVER_SOFTWARE nginx;
fastcgi_param QUERY_STRING $query_string;
fastcgi_param REQUEST_METHOD $request_method;
fastcgi_param CONTENT_TYPE $content_type;
fastcgi_param CONTENT_LENGTH $content_length;
fastcgi_param SCRIPT_FILENAME $document_root $fastcgi_script_name;
fastcgi_param SCRIPT_NAME $fastcgi_script_name;
fastcgi_param REQUEST_URI $request_uri;
fastcgi_param DOCUMENT_URI $document_uri;
fastcgi_param DOCUMENT_ROOT $document_root;
fastcgi_param SERVER_PROTOCOL $server_protocol;
fastcgi_param REMOTE_ADDR $remote_addr;
fastcgi_param REMOTE_PORT $remote_port;
fastcgi_param SERVER_ADDR $server_addr;
fastcgi_param SERVER_PORT $server_port;
fastcgi_param SERVER_NAME $server_name;

fastcgi_param REDIRECT_STATUS 200;

```

我们启动 Nginx 时，经常会遇到 `no input file specified` 的问题，其实这是由于我们的 `nginx.conf` 文件没有指定 `$document_root` 变量，所以我们将下面这行内容改动一下（另外一种方法就是正确指定此变量）：


```
fastcgi_param SCRIPT_FILENAME    $document_root $fastcgi_script_name;
```

将其修改如下:

```
fastcgi_param SCRIPT_FILENAME    /data/htdocs/www $fastcgi_script_name;
```

(7) 启动 Nginx, 命令如下:

```
/usr/local/webserver/nginx/sbin/nginx
```

以后每次更改 Nginx 配置文件想重新启动 Nginx 时, 都可以平滑地启动 Nginx, 这也是 Nginx0.8.x 增加的新特征, 命令如下:

```
/usr/local/webserver/nginx/sbin/nginx -s reload
```

(8) 编写每天定时切割 Nginx 日志的脚本, 创建脚本文件 /usr/local/webserver/nginx/sbin/cut_nginx_log.sh, 文件内容如下:

```
#!/bin/bash
# This script run at 00:00
# The Nginx logs path
logs_path="/data/logs"
mkdir -p ${logs_path} $(date -d "yesterday" +%Y)/$(date -d "yesterday" +%m)/`
mv ${logs_path}access.log ${logs_path}$(date -d "yesterday" +%Y)/$(date -d "yesterday"
+ "%m")/access_$(date -d "yesterday" +%Y%m%d).log
/usr/local/webserver/nginx/sbin/nginx -s reload
```

设置 Crontab, 每天零点切割 Nginx 访问日志。/etc/crontab 文件新增的内容如下:

```
00 00*** /bin/bash /usr/local/webserver/nginx/sbin/cut_nginx_log.sh
```

7. 优化 Linux 内核

我们可以编辑 /etc/sysctl.conf, 新增内容如下:

```
net.ipv4.tcp_max_syn_backlog = 65536
net.core.netdev_max_backlog = 32768
net.core.somaxconn = 32768
net.core.wmem_default = 8388608
net.core.rmem_default = 8388608
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
net.ipv4.tcp_timestamps = 0
net.ipv4.tcp_synack_retries = 2
net.ipv4.tcp_syn_retries = 2
net.ipv4.tcp_tw_recycle = 1
#net.ipv4.tcp_tw_len = 1
net.ipv4.tcp_tw_reuse = 1
net.ipv4.tcp_mem = 94500000 915000000 927000000
net.ipv4.tcp_max_orphans = 3276800
net.ipv4.ip_local_port_range = 1024 65535
```

使配置立即生效, 命令如下:

```
/sbin/sysctl -p
```

8. 将相关软件都配置成自启动模式

我们将一些内容添加进/etc/rc.local 文件中, 如下所示:

```
ulimit -SHn 65535
/usr/local/webserver/php/sbin/php-fpm start
/usr/local/webserver/nginx/sbin/nginx
/usr/local/webserver/mysql/bin/mysqld_safe --defaults-file=/data/mysql/3306/my.cnf
```

基本的安装过程到这里就结束了。通过对线上项目相当长时间的观察, 我们也发现, Nginx 作为 Web 服务器, 在高并发的情况下还是非常稳定的。我也推荐大家将其应用于高并发的 Web 环境中, 尤其是那种日 PV 百万级以上的 Web 网站。

2.9.3 在 Centos5.5 下搭建 Java 运行环境

Tomcat 很受广大程序员的喜欢, 因为它运行时占用的系统资源小, 扩展性好, 支持负载均衡与邮件服务等开发应用系统常用的功能。而且它还在不断地改进和完善中, 任何一个感兴趣的程序员都可以更改它或在其中加入新的功能。

Tomcat 是一个轻量级的应用服务器, 在中小型系统和并发访问用户不是很多的场合下被普遍使用, 是开发和调试 JSP 程序的首选。对于一个初学者来说, 可以这样认为, 当在一台机器上配置好 Apache 服务器时, 可利用它响应对 HTML 页面的访问请求。实际上 Tomcat 部分是 Apache 服务器的扩展, 但它是独立运行的, 所以当你运行 Tomcat 时, 它实际上是作为一个独立的进程单独运行的。

Apache Tomcat 7.x 是目前的发展焦点, 它在汲取了 Tomcat 6.0.x 优点的基础上, 实现了对于 Servlet 3.0、JSP 2.2 和 EL 2.2 等特性的支持。除此以外的其他改进如下:

- Web 应用内存溢出侦测和预防。
- 增强了管理程序和服务器管理程序的安全性。
- 一般的 CSRF 保护。
- 支持 Web 应用中外部内容的直接引用。
- 重构 (connectors, lifecycle) 及对很多核心代码的全面梳理。

Nginx 与 Tomcat 整合后, 有以下优点:

- Nginx 与 Tomcat 的兼容性很好, 静态图片、JS、CSS、Flash 等由 Nginx 来处理, 速度较快, 而文件扩展名为 .jsp、.do 的请求, 则由 Tomcat 来处理。
- 将 Tomcat 默认的 8080 端口改为了 80, 这样我们在输入项目名时不需要带上 8080 端口。
- Nginx 本身的 upstream 模块可以负载均衡后面的 Tomcat 服务器, 这样提高了整个系统的高可用性。

我的线上服务器应用环境为: 64bit Cents5.5 + JDK1.6 + Nginx0.8.46 + Apache-Tomcat7.0.10。整个配置过程如下:

安装 JDK1.6 + Tomcat7.0.10, 截至本文写完时, Tomcat 7.0.10 是当前最为稳定的版本, 不过大家可以用当前最新的版本来测试。

JDK 可以在 Java 的官方网站下载: <http://java.sun.com/javase/download/>。

1) JDK 的安装, 如下所示:

```
chmod +x jdk-6u24-linux-x64.bin
./jdk-6u24-liux-x64.bin
```

2) 配置 Java 运行环境, 如下所示:

```
mv jdk-6u24 /usr/local/jdk
```

我们用 vim 编辑/etc/profile 文件, 在其后添加如下内容:

```
JAVA_HOME="/usr/local/jdk"
CLASS_PATH="$JAVA_HOME/lib:$JAVA_HOME/jre/lib"
PATH=".: $PATH: $JAVA_HOME/bin"
CATALINA_HOME="/usr/local/tomcat"
export JAVA_HOME CATALINA_HOME
export LC_ALL="zh_CN.GB18030"
```

执行如下命令让其立即生效:

```
source/etc/profile
```

3) JDK 安装成功之后, 再安装 Tomcat, 如下所示:

```
tar zxvf apache-tomcat-7.0.10.tar.gz
mv apache-tomcat-7.0.10 /usr/local/tomcat
```

4) 配置 Tomcat, 如下所示:

```
cp -rf /usr/local/tomcat/webapps/* /data/htdocs/www
```

这里要执行 cp -rf 是因为原来 Tomcat 是将其 webapps 作为根的, 它下面的实例也均是以其作为根的, 我们在后面要更改此目录为/data/htdocs/www, 所以将其实例也完全拷贝过去。

编辑/usr/local/tomcat/conf/server.xml, 将 webapps 改为/data/htdocs/www。

编辑/usr/local/webserver/nginx/conf/nginx.conf, 细节方面大家自己处理, 这里就不再重复了。

以下文件直接取自我的线上环境, 内容如下:

```
user www www;
worker_processes 8;
error_log /usr/local/webserver/nginx/logs/nginx_error.log crit;
pid /usr/local/webserver/nginx/nginx.pid;
worker_rlimit_nofile 65535;
events
{
    use epoll;
    worker_connections 65535;
}
http
{
    include mime.types;
    default_type application/octet-stream;
    charset utf-8;
    server_names_hash_bucket_size 128;
    client_header_buffer_size 32k;
    large_client_header_buffers 4 32k;
    client_max_body_size 8m;
    sendfile on;
    tcp_nopush on;
```

```

keepalive_timeout 60;
tcp_nodelay on;
server_tokens off;
client_body_buffer_size 512k;
proxy_connect_timeout 5;
proxy_send_timeout 5;
proxy_read_timeout 60;
proxy_buffer_size 16k;
proxy_buffers 4 64k;
proxy_busy_buffers_size 128k;
proxy_temp_file_write_size 128k;
gzip on;
gzip_min_length 1k;
gzip_buffers 4 16k;
gzip_http_version 1.1;
gzip_comp_level 2;
gzip_types text/plain application/x-javascript text/css application/xml;
gzip_vary on;

upstream tomcat_server{
    server 127.0.0.1:8080;
}

server
{
    listen 80;
    server_name www.cn7788.com;
    index index.html index.htm index.jsp default.jsp index.do default.do;
    root /data/htdocs/www;
    if (-d $request_filename)
    {
        rewrite ^/(.*) ([^/]) $http:// $host/ $1 $2/ permanent;
    }

    location ~ \.(jsp|jspx|do)?$ {
        proxy_set_header Host $host;
        proxy_set_header X-Forwarded-For $remote_addr;
        proxy_pass http://tomcat_server;
    }

    location ~ \.(gif|jpg|jpeg|png|bmp|swf) {
        expires 30d;
    }

    location ~ \.(js|css) {
        expires 1h;
    }

    location /nginxstatus {
        stub_status on;
        access_log on;
    }
}

```



```

    }
    log_format  wwwlogs  '$remote_addr - $remote_user[ $time_local] "$request" '
        '$status $body_bytes_sent "$http_referer" '
        '" $http_user_agent" $http_x_forwarded_for';
    access_log  /data/logs/www_tomcat.log  wwwlogs;
}
}

```

平滑重启 Nginx 命令，命令如下：

```
/usr/local/webserver/nginx/sbin/nginx -s reload
```

开启 Tomcat，命令如下：

```
/usr/local/tomcat/bin/startup.sh
```

关闭 Tomcat，命令如下：

```
/usr/local/tomcat/bin/shutdown.sh
```

可能有许多朋友跟我一样，主要工作用的是 PHP + MySQL。所以下面也附上 Tomcat7.0.10 的目录结构，方便大家理解以上配置。

Tomcat 的目录结构如下所示：

- /bin：存放 Windows 或 Linux 平台上启动和关闭 Tomcat 的脚本文件。
- /conf：存放 Tomcat 服务器的各种全局配置文件，其中最重要的是 server.xml 和 web.xml。
- /doc：存放 Tomcat 文档。
- /temp：Tomcat 的临时文件存放路径。
- /lib：存放 Tomcat 服务器所需的各种 JAR 文件。
- /logs：存放 Tomcat 执行时的日志文件。
- /webapps：Tomcat 的主要 Web 发布目录，默认情况下把 Web 应用文件存放于此目录中。
- /work：存放 JSP 编译后产生的 class 文件。

然后测试 Java 能否正常运行，如下所示。

进入 /data/htdocs/www/root，编辑 hello.jsp 代码，内容如下：

```

<%
out.println("hello world");
%>

```

接着我们写一个测试 mem 的代码，代码 mem.jsp 的内容如下：

```

<%
Runtime lRuntime = Runtime.getRuntime();
out.println("*** BEGIN MEMORY STATISTICS*** <br/>");
out.println("Free Memory:" + lRuntime.freeMemory()/1024/1024 + "M<br/>");
out.println("Max  Memory:" + lRuntime.maxMemory()/1024/1024 + "M<br/>");
out.println("Total Memory:" + lRuntime.totalMemory()/1024/1024 + "M<br/>");
out.println("Available Processors : " + lRuntime.availableProcessors() + "<br/>");
out.println("*** END MEMORY STATISTICS*** ");
%

```

我们尝试在 IE 或火狐下输入 `http://203.93.236.146/mem.jsp` 看能否正常运行。成功运行后结果如下：

```
*** BEGIN MEMORY STATISTICS***  
Free Memory:19M  
Max Memory:878M  
Total Memory:44M  
Available Processors :4  
*** END MEMORY STATISTICS***
```

2.10 小结

本章以 FreeBSD8.1 为平台，详细描述了 FreeBSD8.1 x86_64 在企业中的部署应用。在后面的章节中有许多重要的服务（比如 bind、postfix）中虽没有提及用 FreeBSD8 或 FreeBSD8.1 来搭建，这不是说明它们不能采用，主要是我个人习惯问题（我较喜欢采用 CentOS5.5 来架设）。而且以上许多服务的语法，比如 Samba、rsync、vsftpd，还有 SVN 及 Git 服务器的语法配置跟 CentOS5.5 下的很类似，只不过系统平台不一样而已，喜欢用 CentOS5.5 的朋友也可以在其下依此部署这些应用服务。相对于其他开源系统来说，FreeBSD 系列的特点是稳定、方便、严谨，尤其是在开发环境部署 PHP5 + MySQL + Apache 及 jail（ezjail）虚拟机等应用服务时，这些优点更为突出，这一点希望大家在工作中进一步体会。我也希望大家能在熟悉 CentOS5.5 的前提下，了解 FreeBSD8.1 的基本配置和操作，这会给大家今后的工作带来帮助。



第 3 章 Linux服务器虚拟化

- 3.1 在 Windows Server 2003 下安装 VMware GSX Server
- 3.2 用 Windows 2003 + VMware Server 搭建 64 位系统测试环境
- 3.3 在 CentOS5.6 x86_64 下安装 Xen 虚拟机
- 3.4 XEN 在生产环境下的应用
- 3.5 Citrix XenServer5.6 虚拟机试用手记
- 3.6 小结

在内网开发环境中，Linux 下的 Xen 虚拟机和 FreeBSD 下的 jail 虚拟机已作为开发服务器使用很久了，Xen 的高效和稳定让我们印象深刻。Xen 是可用于 Linux 内核的一种虚拟化技术，我们可以在原有 Linux 环境中安装并测试新的升级，而不必担心原有的系统被破坏。对于企业而言，如何尽可能减少 IT 设备的采购成本和人力成本是一件十分重要的事情，也是系统管理员的工作职责之一。Xen 是 Linux 系统自带的免费且开源的虚拟化软件，其性能也不错，因此，我们推荐使用它。尽管它自身也有着许多不足，但我们可以充分了解其特点的前提下，在使用的时候尽量扬长避短。

本章会向大家介绍一些我在工作中常用的虚拟化软件，它们是 VMware GSX Server（也包括 GSX 的升级版 VMware Server）、Centos 下的 Xen 及思杰的 XenServer。在本章中，有关 Xen 在生产环境中的应用（3.3 节）的内容是我的朋友曹亚孟^①（System Admin）先生从他这几年使用 Xen 虚拟化的实际工作中总结而来的，是非常宝贵的一线生产环境下的技术资料。在这里对他无私奉献的精神表示衷心的感谢和由衷的敬佩。

3.1 在 Windows Server 2003 下安装 VMware GSX Server

前段时间因为学习的需要配置了一台性价比比较高的机器，CPU 选择了双核速龙。当时有学习 Windows Server 2003 的需求，所以我特地安装了 32 位的 Windows Server 2003 企业版，运行了一段时间后发现它相当稳定，而且占用系统资源非常少，系统 uptime 也非常低。当准备通过光盘安装 FreeBSD8 来学习一下时，突然闪现一个念头，能不能直接用宿主机安装另外一个系统呢？随后我如此操作并取得了成功，下面我就和大家分享一下具体的操作过程。

这里首先说明一下机器的配置。

- CPU：速龙 64 X2 5000 +
- 内存：威刚 DDR2 800 2GB
- 主板：昂达 N61P
- 网卡和显卡：主板自带

我的需求如下，相信不少人也和我有同样的需求：

- 要保证宿主机稳定运行。
- 虚拟系统要随着宿主机的启动而启动。
- 虚拟系统也要能稳定运行。

以前在公司用的开发环境是 Windows Server 2003 + VMware GSX Server v3.2.1，效果相当好，虚拟出来的系统在开发中都很稳定。所以我就在自己的机器上也安装了以上版本的 VMware GSX Server v3.2.1。

VMware GSX Server 安装虚拟机的过程与 VMware WorkStation6 一样，非常简单。这里说一下大家比较关心的网络问题：如果是在小区或公司内上网，即有 DHCP 服务器，则直接用 bridged（物理）连接即可；如果是用 ADSL 或无线网卡上网，也很方便，直接用 NAT 共享上网即可。FreeBSD8 对网络的依赖性很大，如果网络安装通畅，以后学习和工作就非常方便了。这里值得说一下的是，如

① 5 年系统运维经验，曾经运营过线上教育网站、工业级生产系统和门户网站等。能熟练使用 Linux 下常用的系统服务、Web 服务、数据库服务并有深刻的见解。从 2009 年开始研究虚拟化系统在生产环境中的应用，积累了丰富的实战经验。

果是 ADSL 共享上网，用 Windows Server 2003 作宿主机非常方便（FreeBSD8.1 直接选择 NAT 共享即可），如果是重新安装双系统，在 FreeBSD 下建立 ADSL 拨号还是很麻烦的。

这里说一下其中的难点：在安装完 FreeBSD8 后，如何让其随着 Windows Server 2003 的启动而启动，以及随着它的关机而关机呢？我按照以前在公司的配置方法很轻松地就实现了，即依次打开 Virtual Machine Settings→Options→Startup/Shutdown 选项，然后依次选择 Run this virtual machine as→Locate system account。下面的 on host startup 选择 Power on virtual machine，On host shutdown 则选择 Power off virtual machine。具体设置如图 3-1 所示。

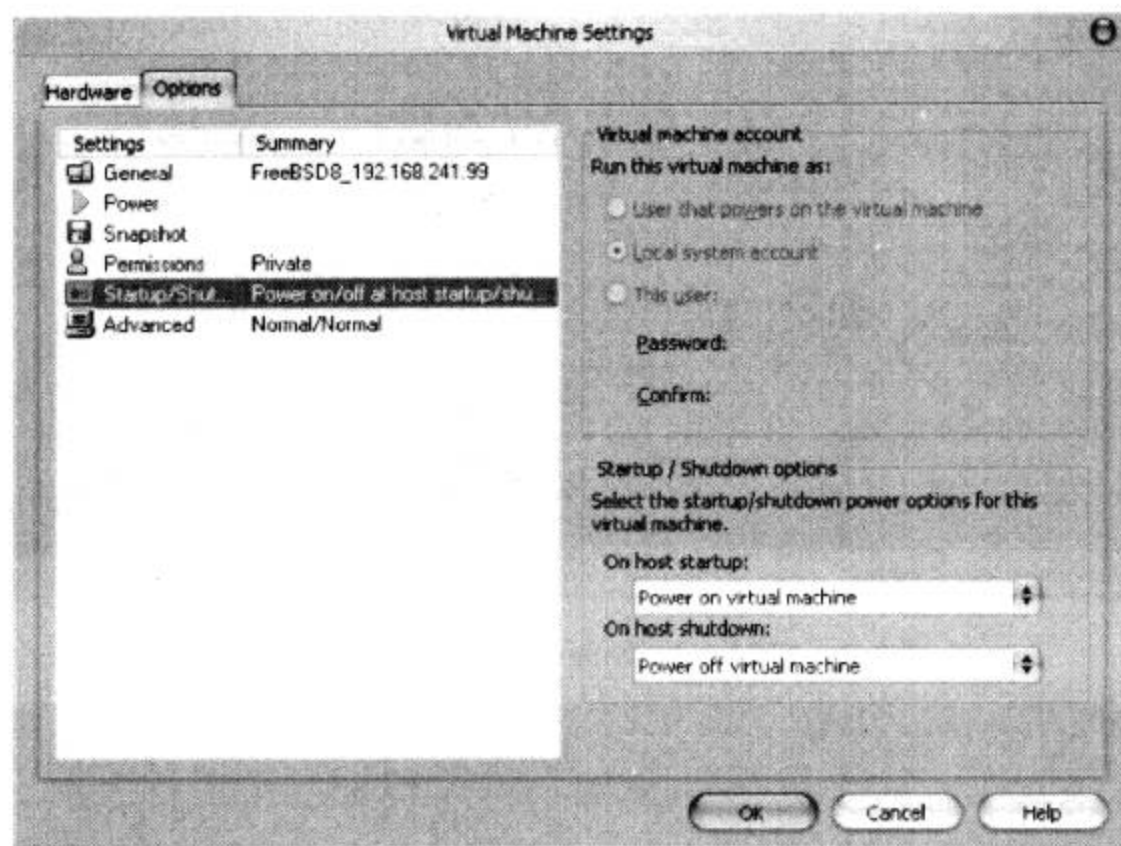


图 3-1 将 VMware GSX Server 的 VM 机器设置成自启动状态

VMware GSX Server 本身就带有远程控制终端，很方便，它提供了 VMware Virtual Machine Console 来管理远程服务器的虚拟系统。如果有多台 VMware GSX Server 就更方便，比 Windows 自带的“远程桌面”（mstsc）和 Xmanager3.0 企业版方便多了，而且它们还可以作为辅助工具。VMware GSX Server 的 console 管理工具如图 3-2 所示。

为了达到精简的目的，我给 FreeBSD8.1 系统划分的硬盘大小为 2GB（最小化安装），内存为 96MB。VMware GSX Server 就是好用，后期无论是你升级还是做别的工作，FreeBSD8 的磁盘空间都只需要 2GB。大家可能会怀疑 2GB 小了，其实已经足够，我在安装完 FreeBSD 后，还通过光盘升级安装了 src ports，并且升级了最新的“ports: portsportsnap fetch update”，升级完毕后磁盘空间仍有富余，用来学习足够了。当然，如果大家的硬盘空间足够，分配 20GB 空间安装一个桌面也未尝不可。我们可以看一下 VMware GSX Server 下 FreeBSD8 虚拟机的硬盘使用情况，如图 3-3 所示。

在工作中的好处就是，FreeBSD 在后台启动（GSX 在后台以 vmware-vmx 的服务方式启动），你完全感觉不到你的机器上还存在另外一个系统，只是在需要时才会调用。可能有的朋友比较担心内存占用情况，我给系统划分的内存为 96MB，下面给大家看一下 360 安全卫士的进程管理器及 Windows Server 2003 自带的资源管理器 taskmgr 所显示的内存使用情况，如图 3-4 和图 3-5 所示。当然，如果大家的机器内存足够，这个问题完全可以忽略。

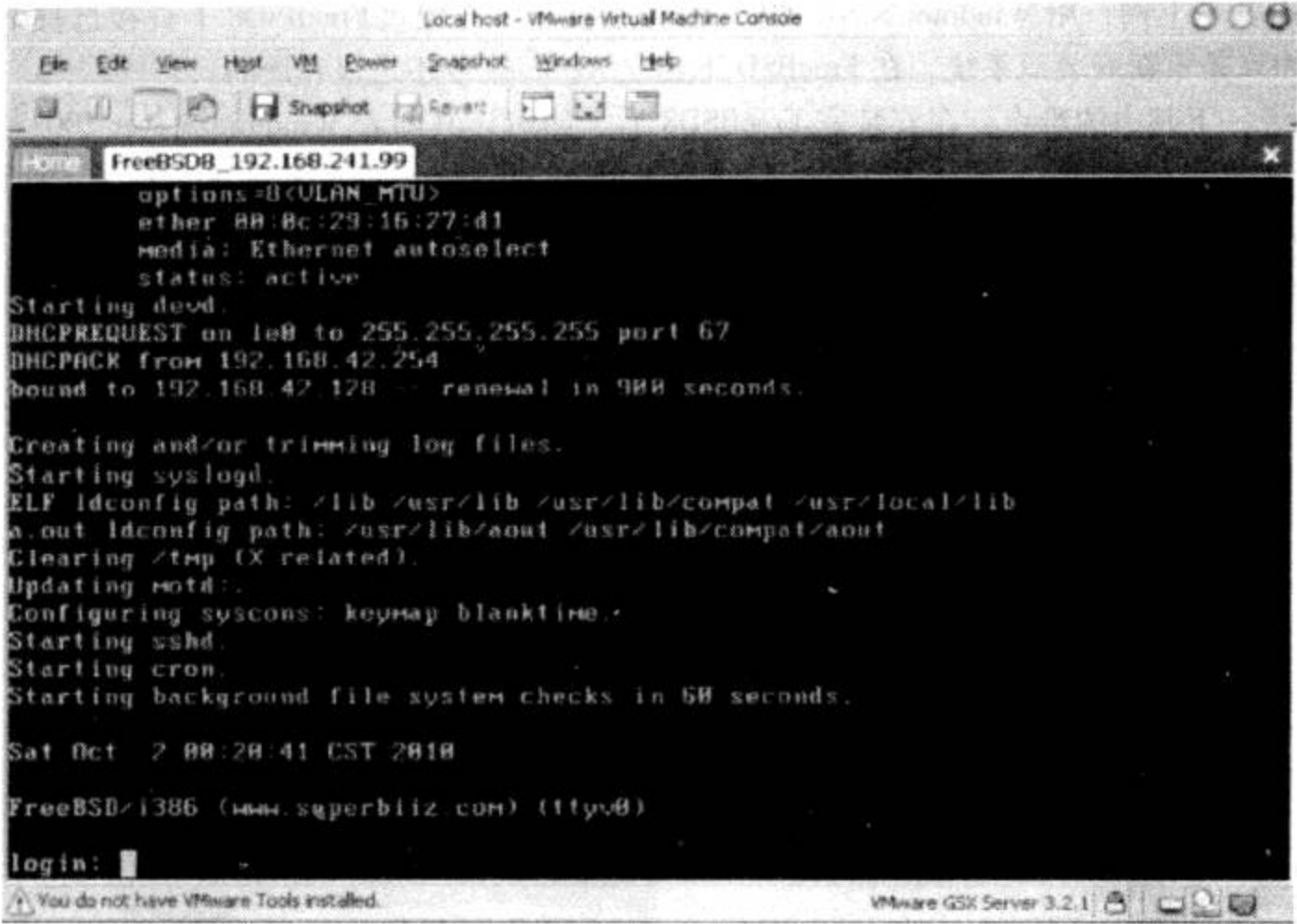


图 3-2 VMware GSX Server 的 console 管理工具

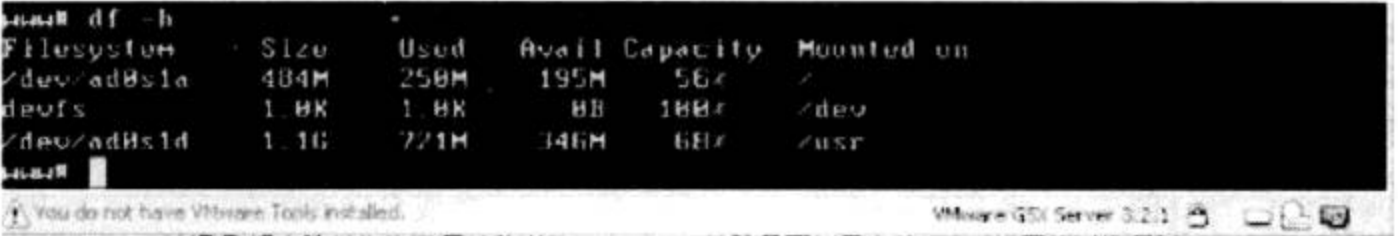


图 3-3 FreeBSD8 虚拟机的硬盘使用情况



图 3-4 360 进程管理器显示系统内存占用情况



图 3-5 系统资源管理器显示详细信息

这里有一点要说明一下：在 Windows XP SP3 下安装 VMware GSX Server v3.2.1 不仅很麻烦，而且安装后使用起来也不方便，用起来有点卡，很不流畅，完全没有在 Windows Server 2003 下使用的那种感觉。如果确实想在 Windows XP SP3 下安装，建议安装其升级版本 VMware Server1.01 或更高级版本，这一点我在第 2 章已介绍，这里就不再赘述了。在工作和娱乐中我们可以用 Windows Server 2003 浏览新闻或观看视频，同时还可以用 Xmanager3 或 GSX Server 自带的工具 VMware Virtual Machine Console 连接到 FreeBSD8 下去学习。

大家按以上方法安装了 FreeBSD8 以后，平时可以练习一下 vim、sed 及 awk 和 shell 的用法，还可以熟悉一下 FreeBSD8 的命令，它与 Centos5.5 还是有区别的。如果大家想安装 64 位的系统，可以使用 VMware GSX Server 的升级版本 VMware Server1.01。我在 32 位的 Windows XP SP3 系统上先安装了 VMware Server1.01 后安装了 64 位的 Centos5.5，运行也非常流畅，有兴趣的朋友也可以尝试。

3.2 用 Windows 2003 + VMware Server 搭建 64 位系统测试环境

虽然很多开发人员都没有自己的工作服务器，但大家可以通过自己的组装机或组装机服务器来进行 64 位系统的测试。曾经我所在的公司的许多环境中都需要在 64 位系统下进行代码的测试和环境的部署，所以我一直在尝试搭建 64 位的虚拟机环境。但是，由于目前所有的服务器不是用于线上的生产环境，就是用于内网的开发测试，可以使用的服务器只有组装机了。在这些机器上安装并测试 VMware ESXi 时失败了（兼容性相对更好的 XenServer5.6 对 FreeBSD 的支持也不是太好，有的组装机无法安装），所以我尝试了 Windows 2003 x86_64 + VMware Server1.01（VMware Server 是 VMware GSX Server 的升级版本）的方法，效果比较好，很适合作为测试环境，特向大家推荐。

我的组装服务器硬件配置清单如表 3-1 所示。

表 3-1 组装服务器配置清单

硬件名称	规格型号	硬件名称	规格型号
CPU	INTEL i5-760	内存	金士顿 DDR3 1333 2G * 4
网卡	主板自带	硬盘	ST SATA2 1.5T
主板	华硕 P7F-X		

为了方便管理，我在每台服务器上都安装了 Remote Admin3.4，使用它主要是出于如下考虑：

- 进行远程控制时比远程桌面更方便和快捷，尤其适用于网速慢的情况。
- 不需要网上邻居就能方便地上传和下载。
- Remote Admin3.4 能支持双显卡，这个功能很强大。
- 不会像 Windows 的远程桌面那样出现超出最大连接数的问题，事实上这个问题可以通过 Remote Admin3.4 很轻松地解决。

其效果图如图 3-6 所示。我们在工作中使用的 Windows XP 机上安装 Radmin View3.4 就可以方便地控制了。

我在服务器上安装了 6 台 64 位的 Windows Server 2003 R2、FreeBSD8.1 和 Centos5.5，用于程序调试和分布式实验，感觉很好用，服务器的负载也不是太大，效果如图 3-7 所示。

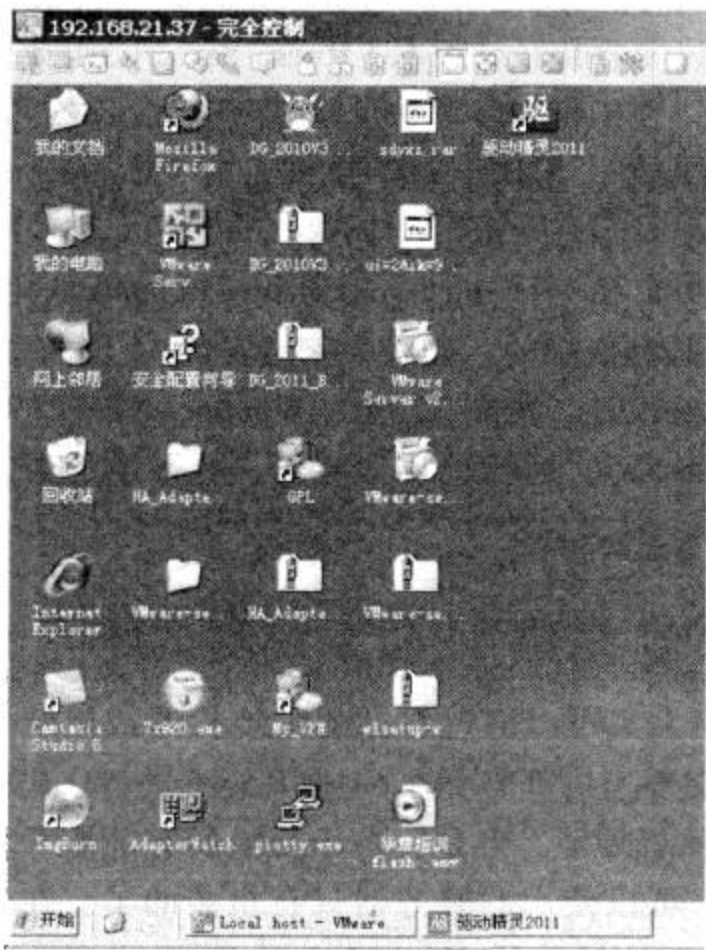


图 3-6 Remote Admin3.4 远程控制界面



图 3-7 Windows Server 2003 任务管理器界面

整个安装过程很简单，这里只说一下其中的难点：安装完 FreeBSD8 或 Centos5.5 等其他虚拟机后，如何让其随着 Windows Server 2003 的启动而启动、关机而关机呢？这一点已经在 3.1 节详细讲

解过，这里不再重复。

VMware Server 端的整个配置过程就结束了，相信许多朋友和我一样是在 Windows XP SP3 或 Win7 环境下办公的，我们可以在自己的工作机上安装“VMware Server Console”工具，于是可以像在本机操作 VMware Server 一样方便地进行测试工作（工作机制类似于远程桌面）了，以免在机房苦苦守候等候。

3.3 在 Centos5.6 x86_64 下安装 XEN 虚拟机

3.3.1 XEN 在 Centos5.6 x86_64 下的安装步骤

1. 安装 XEN 前的准备工作

首先介绍一下安装 XEN 的机器的配置，它的配置很普通，如下所示。

- CPU：速龙 64X2 5000 +
- 内存：威刚 DDR2 800 2GB
- 主板：昂达 N61P
- 硬盘：希捷 IDE 40G
- 系统：Centos5.5 x86_64

速龙 CPU 是基于 64 位架构的，性价比非常高。硬盘用的是以前被淘汰的一块老 IDE 硬盘，只有 40GB 大小，由于盘上已有 Windows XP SP3 的系统，所以我特地划分了 18GB 的 Free 空间来安装 Centos5.6 x86_64。

在安装 XEN 之前我们先检查一下 CPU 是否支持 XEN 虚拟化，命令如下：

```
egrep '(vmx|svm)'/proc/cpuinfo
```

如果什么结果都不显示，则表示 CPU 是不支持 XEN 虚拟化的，如下显示表明是支持的：

```
flags                :fpu tsc msr pae cx8 apic mtrr cmov pat clflush mmx fxsr sse sse2 ht syscall
nx mmxext fxsr_opt lm 3dnowext 3dnow pni cx16 lahf_lm cmp_legacy svm cr8_legacy misalignsse
flags                :fpu tsc msr pae cx8 apic mtrr cmov pat clflush mmx fxsr sse sse2 ht syscall
nx mmxext fxsr_opt lm 3dnowext 3dnow pni cx16 lahf_lm cmp_legacy svm cr8_legacy misaligns
```

2. XEN 的安装过程

可以用以下命令来安装 XEN 软件：

```
yum -y install kernel-xen xen
```

顺利安装完 Centos5.6 后，要修改/etc/grub.conf 文件，让其采用新的内核，/etc/grub.conf 文件的内容如下：

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE:  You have a /boot partition.  This means that
#           all kernel and initrd paths are relative to /boot/, eg.
#           root (hd0,6)
#           kernel/vmlinuz-version ro root = /dev/VolGroup00/LogVol00
```

```
#          initrd/initrd-version.img
#boot = /dev/hdb
default=0
timeout=5
splashimage = (hd0,6)/grub/splash.xpm.gz
hiddenmenu
title CentOS (2.6.18-238.12.1.el5xen)
    root (hd0,6)
    kernel/xen.gz -2.6.18-238.12.1.el5
    module/vmlinuz -2.6.18-238.12.1.el5xen ro root = /dev/VolGroup00/LogVol100
    module/initrd -2.6.18-238.12.1.el5xen.img
title CentOS (2.6.18-238.el5)
    root (hd0,6)
    kernel/vmlinuz -2.6.18-238.el5 ro root = /dev/VolGroup00/LogVol100
    initrd/initrd -2.6.18-238.el5.img
title Other
    rootnoverify (hd0,0)
    chainloader +
```

3. 开始安装 XEN 虚拟机

下面要开始安装 XEN 虚拟机了，安装前要做好准备工作，即搭好 httpd 环境，因为我们需要 httpd 服务，这很容易实现，命令如下：

```
yum-y install httpd && service httpd start
```

另外，记得新建一个目录供 XEN 安装虚拟机使用，我这里建立的是 vm。我准备安装两个 XEN 虚拟系统，一个为 Centos5.6 x86_64，另一个为 Centos5.0 i386，所以我将其光盘文件分别用 cp -a 命令分别复制到 /var/www/html/centos56 和 /var/www/html/centos5 目录下了。这些都是 Linux 下的简单操作，大家应该都很熟悉。如果只有 iso 文件，可以直接在 Centos 下用 mount 命令挂载，命令如下：

```
mount -t iso9660 -o loop,user download.iso /isoimage
```

(1) 安装 Centos5.6 x86_64 XEN 虚拟机，安装命令如下：

```
virt-install -n vm0 -r 256 -f /vm/vm01.img -s 4 --nographics -p -l http://192.168.1.119/centos56
```

(2) 安装完 vm0 的 XEN 虚拟机后，再安装一个名为 vm1 的 XEN 虚拟机，命令如下：

```
virt-install -n vm1 -r 256 -f /vm/vm02.img -s 4 --nographics -p -l http://192.168.1.119/centos5
```

i386 系统很顺利地就安装成功了，可以用 xm list 命令来查看一下，如下所示：

Name	ID	Mem (MiB)	VCPUs	State	Time (s)
Domain-0	0	1193	2	r-----	114.0
vm0	3	256	1	-b-----	7.6
vm1	2	511	1	-b-----	18.

XEN 虚拟机常用的命令如下。

- `xm list`: 查看当前机器里的全部虚拟机列表。
- `xm create xxx`: 启动名字为 `xxx` 的虚拟机。
- `xm shutdown xxx`: 关闭名字为 `xxx` 的虚拟机。
- `xm reboot xxx`: 重启名字为 `xxx` 的虚拟机。
- `xm pause xxx`: 暂停名字为 `xxx` 的虚拟机。
- `xm resume xxx`: 继续运行名字为 `xxx` 的虚拟机。

需要管理登录的时候使用如下命令:

```
xm console xxxx
```

可以像在本机一样操作虚拟机。如果要退出到 Domain-0 上, 按 [Ctrl + I] 键就行了。

3.3.2 XEN 虚拟机的优势

我以前在研究虚拟化时进入了一个误区, 就是想用一个虚拟机完美地虚拟出我需要的所有系统, 比如说 Windows Server 2003、FreeBSD8, 以及 Centos 和 Debian, 所以我一直都在研究 VMware ESX 系列。后来我发现这种思想是错误的, 这也是我决定花时间和精力在 XEN 上面的原因。XEN 的优势有如下 3 点:

- FreeBSD 下 jail (ezjail) 的性能出乎我的意料, 它的使用范围很小, 只能虚拟出 FreeBSD 系统, 但这已足够应付工作了。以往在用 XEN 时, 总觉得 XEN 虚拟出的 Windows 系统很慢, 其实完全没有必要这样做。为什么要用 Linux 的 XEN 去虚拟 Windows 呢? 在工作中我们完全可以用 Linux 的 XEN 虚拟 Linux, 只要能满足工作要求就行, 大家可以看一下 XEN 作为 VPS 的使用率。
- 大家都知道, XenServer 系列是基于 XEN 发展而来的, 它拥有 xenServer 的优秀性能。
- 各种 Linux 系统都自带了 XEN 虚拟机, 而且是永久免费的, 大家不需要担心版权或 license 的问题。

XEN 利用模板功能 5 分钟就能克隆出一台 XEN 虚拟机来, 而且在大规模的虚拟机集群部署上, XEN 也非常方便, 短时间就能部署 200 台以上的 XEN 虚拟机集群。

3.4 XEN 在生产环境下的应用

3.4.1 XEN 虚拟化的基本概念

1. 虚拟化的概念

虚拟化的前身是分时操作系统, 它能在操作系统层面上分时和分片, 而 XEN 则是在操作系统层和硬件层之间进行分片的, 并且每个片上面都托着一个 Guest OS。虚拟化的前身是模拟, 大家可以想象一下红白机和街机的模拟器程序, 被模拟的对象完全意识不到自己是运行在 x86 架构的服务器上的, 它们把数据交给 CPU 运算以及从内存中存取数据等操作都是软件模拟出来的。不过, 这种低效率的模拟没有管理被模拟对象的能力, 也没有必要这么快。

可以把虚拟化技术理解为在硬件层和虚拟机之间增加了一个 VMM 层, 由该层对虚拟机和硬件

进行管理,如图 3-8 所示。VMM 要完成对硬件的识别和管理,还要对虚拟机操作系统的资源进行分配和约束。VMM 层就像一个透明的代理服务,如果这个层很“厚”,VMM 管理虚拟机的灵活性必然会大大提高,但管理的开销也会降低 Guest OS 的执行效率;如果这个层很“薄”,VMM 的管理功能就会很薄弱,而 Guest OS 的执行效率就会大大增加。

2. 3 种不同的虚拟化类型

在 x86 架构下,处理器内的指令可分为不同的 Ring,其中 Ring 0 的指令只有操作系统才能发出。但这会导致出现一些问题,比如 Poweroff 是个很正常的指令,但虚拟机中的 OS 并未意识到自己是存活在虚拟硬件中的,

所以它发出的关机指令从理论上来说也会被 CPU 执行,从而会关闭整个硬件系统,造成灾难。我们可以用 3 种思路来解决这个问题,即虚拟化的 3 种类型。

(1) 全虚拟化

在这种思路中,虚拟机的 OS 不会做任何更改,可以像正常服务器一样发送任何指令,但 VMM 会对虚拟机指令做全程过滤。比如说 Guest OS 发出 Poweroff 指令,VMM 会将该指令改为仅仅关闭当前虚拟机的指令。不过,实现这种方法的 VMM 层技术很复杂,VMM 本身就更难于编写了,巨大的管理(全程监控/过滤)开销会耗费很多硬件资源。

(2) 半虚拟化

在这种思路中,要对虚拟机的内核做一些更改,虚拟机可意识到自己的环境,当发出 Poweroff 指令时仅要求 VMM 关闭本虚拟机使用的资源,而非自作主张地关闭主板电源。这种方法很好,但并不是所有的操作系统都允许你修改它的内核,比如说 Windows。

(3) 硬件虚拟化

这种思路依靠 CPU 本身的功能来实现,比如说 Inter VT-X 或 VT-x 与 AMD-V,它们通过芯片辅助技术将虚拟 OS 的 Ring 0 指令自动调节为 Ring 1,或者在 0 级下面新定义了一个 Ring-1,VMM 的指令则运行在 -1 级别,该级别行使普通服务器 Ring 0 的权利。

全虚拟化的代表作是 VMware Workstation,它很灵活但效率太低;硬件虚拟化没有相关的 CPU 就没法实现。如果在普通的 x86 下用 XEN Linux 做测试或生产,性价比最高的选择就是半虚拟化。

3. 查看硬件对虚拟化的支持情况

64 位的 x86 CPU 都支持半虚拟化,32 位的 CPU 只要支持 PAE 技术就支持半虚拟化,如下所示:

```
[root@XEN ~]# cat /proc/cpuinfo | grep flags
flags           :fpu tsc msr pae mce cx8 apic...
```

硬件虚拟化的 CPU flags 是 Intel (vmx) /AMD (svm),如下所示:

```
[root@XEN ~]# cat /proc/cpuinfo | grep flags
```

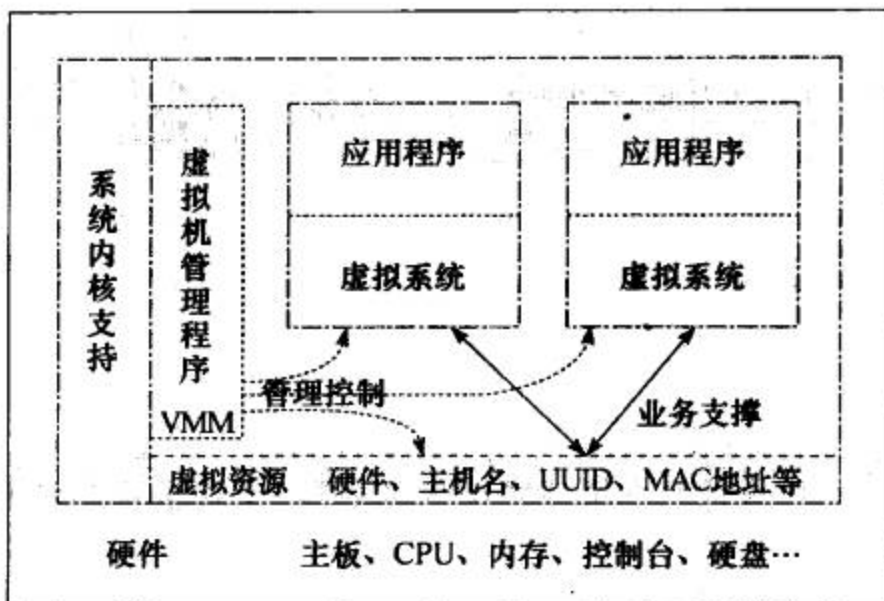


图 3-8 XEN 的工作原理图


```

flags          :fpu tsc msr pae mce cx8 apic  vmx
[root@ XEN ~]# cat /proc/cpuinfo | grep flags
flags          :fpu tsc msr pae mce cx8 apic  svm

```

3.4.2 在 Centos5.5 下安装 XEN 虚拟机

1. 全新安装 XEN

如果在一台没有操作系统的机器上安装 XEN，可以使用 Centos5.5 的安装盘直接安装，与安装普通操作系统的差别在于磁盘分区和组件的选择。具体步骤如下：

(1) 合理地划分磁盘分区，生产环境中的磁盘划分方法如下。

/boot: 256MB (这个分区应该都没异议吧?)

/内存大小 + 20GB (20GB 是保守的正常开销，同内存大小相当的硬盘空间用来存放虚拟机快照)

SWAP: 5GB ~ 10GB (内存总量很大，但随后会限制分配给宿主实体机 1GB ~ 4GB 内存)

/data: 其他所有空间 (系统安装后再将其划分为 LVM 分区，弹性扩充虚拟机的磁盘空间)

(2) 如果是新装系统，则在安装系统时选择下列软件包：

```

Applications  -- Editors
Development   -- Legacy Software Development
               -- Development Libraries
               -- Development Tools
Base System    -- Administration Tools
               -- System Tools
               -- Base

```

可选择安装：

```

               -- Dialup Network Support -- lrzsz
Virtualization -- Virtualization

```

2. 在已有的最小化安装系统上安装 XEN

如果要在已经安装好的系统上安装 XEN，或者安装 RHEL6.0 这类没有 XEN 的系统，需要手动补全 RPM 包。XEN 的详细安装过程如下所示。

(1) 检查这些 RPM 依赖包是否存在，如果不存在，请从系统盘中寻找或 yum 安装，如下所示：

```

rpm -q bridge-utils iproute grub gcc binutils make zlib zlib-devel
rpm -q python python-devel dnsmasq SDL cyrus-sasl-md5 iscsi-initiator-utils
rpm -q XEN-libs libvirt libvirt-python python-virtinst

```

(2) 挂载并进入系统盘的 RPM 包目录，执行下列命令就可以安装 XEN 的相关软件包了。

```

rpm -ivh kernel-xen-2.* kernel-xen-devel-2.* xen-devel* xen-3*

```

3. 系统优化设置

XEN 系统安装完后，还要做一些设置和优化，于是它才能正常工作。

(1) 关闭 SeLinux，设定启动级别为 3，精简系统启动的服务如下：

```
crond lvm2 -monitor network syslog sshd XENd XENdomains
```

(2) 固化宿主机使用的内存，以防止出现内存挤压 Bug。

在实际生产中会发现，XEN 服务本身并不会占用多少系统资源，只要有 1GB 内存就能很平稳地运行 XEN 服务了，但是考虑到 Linux 系统会将许多内存当作 cache 和 buffer 来缓解磁盘 I/O 压力，所以根据内存的总量情况，建议给 XEN 服务器分配 1GB ~ 4GB 的固定内存。

安装了 XEN 以后，grub.conf 的内容应该是下面这样的（版本不同可能会有细微区别）：

```
default=0          #这里一定要确认启用的内核是 XEN 相关内核
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title CentOS (2.6.18-164.el5)
    root (hd0,0)
    kernel/xen.gz -2.6.18-164.el5
    module/vmlinuz -2.6.18-164.el5xen ro root=LABEL=/
    module/initrd -2.6.18-164.el5xen.img
title...
```

如果是在已有的系统上安装 XEN，会保留默认不支持 XEN 的原有内核。假设这台服务器有 32GB 的内存，我们可加上“dom0_mem=4096M”命令，以便设置 XEN 服务器固定使用 4GB 的内存，如下所示：

```
kernel/boot/xen.gz -2.6.18-194.el5 ==> kernel
/boot/xen.gz -2.6.18-194.el5 dom0_mem=4096M
```

配置文件是/etc/xen/xend-config.sxp，请更改 dom-min-mem 的数值，使之与 grub.conf 中的一致。更改的具体数值如下所示：

```
(dom0 -min -mem 256) ==> (dom0 -min -mem 4096)
```

4. 验证安装成功

重启系统后需要依次验证以下内容。

(1) 启动的内核为 xen 内核，命令如下所示：

```
[root@ server-1 ~]# uname -r | grep xen
2.6.18-194.el5xen
```

(2) 当前系统的内存只显示 4GB，但通过 XEN 相关命令看到的内存仍然是 48GB，命令如下所示：

```
[root@ server-1 ~]# free -m
total      used      free      shared    buffers     cached
Mem:       4096      1101       2994         0        177       563
-/+ buffers/cache:      360      3735
Swap:      4094         0       4094
[root@ server-1 ~]# xm info | grep total_memory
total_memory      :49142
```

(3) xend 服务由 xen 控制，安装完成后会自动启动。

```
[root@ server-1 ~]# /etc/init.d/xend status
```

```
xend is running
[root@ server-1 ~]# head/etc/init.d/xend
#!/bin/bash
# xend          Script to start and stop the Xen control daemon.
```

(4) xendomains 服务由虚拟机控制，安装完成后 XEN 会自动启动。

下面命令中的“web001 web002 md001 md002”是正在运行的虚拟机名称。

```
[root@ idctest-1-2 ~]# /etc/init.d/xendomains status
Checking for xendomains:web001 web002 md001 md002[running][ OK ]
[root@ idctest-1-2 ~]# head/etc/init.d/xendomains
#!/bin/bash
# Start/ stop domains automatically when domain 0 boots/ shuts down.
```

5. Domain-0 的硬件管理

待 XEN 安装完后就可以使用一些与 XEN 相关的命令了，其中最重要的是 XM 命令。xm list 命令可以列出现有的虚拟机，默认已经有一个特殊的虚拟机 Domain-0 在运行了，如下所示。

```
[root@ xen ~]# xm list
Name                                ID Mem (MiB) VCPUs State  Time(s)
Domain-0                            0      879      2 r----- 89.2
```

当 xend 进程启动的时候，会根据配置文件和系统现状将一部分资源划入 domain-0 中，然后 domain-0 再把资源分配给虚拟机来使用。它的身份其实就是上文提到的 VMM 的管理接口。比如说，我们在创建新的虚拟机时恶意指定了一个大内存值，xend 和 domain-0 就会跳出来阻止，如下所示：

```
xend.err 'Error creating domain:I need 919552 KiB, but dom0_min_mem
is 262144 and shrinking to 262144 KiB would leave only 765888 KiB free.'
```

至此，XEN 服务器使用默认的配置就已经可以正常工作了。如果还想要进行其他设置，可以去看看/etc/xen/xend-config.sxp。

3.4.3 安装第一台虚拟机（模板机）

1. 可选的安装形式

先说明一下，我的测试环境是半虚拟化的。在硬件虚拟化的环境里，可以用 ISO 文件、CD-ROM 来安装虚拟机，但在半虚拟化环境里，只有以下 3 种方法可安装虚拟机：

- 用 NFS、HTTP、FTP 的方式安装系统。
- 直接挂载被虚拟的磁盘，然后像创建 LFS 一样将系统文件、引导文件都存入虚拟磁盘中。
- 从网上下载他人制作好的迷你系统镜像（已经安装好了基本系统、ssh 等组件）。

这里推荐用第一种方法安装。

2. 准备安装介质

首先要启用 NFS 服务，将光盘目录 NFS 共享给内网，命令如下所示：

```
[root@ XEN ~]# /etc/init.d/portmap start
[root@ XEN ~]# /etc/init.d/nfs start
```

```
[root@ XEN ~]# mkdir -p /mnt/cdrom
```

然后插入要安装的系统光盘，并挂载，命令如下所示：

```
[root@ XEN ~]# mount /dev/cdrom /mnt/cdrom/
[root@ XEN ~]# echo "/mnt/cdrom 192.168.118.0/24 (ro)" >> /etc/exports
[root@ XEN ~]# /etc/init.d/nfs restart
```

另外，如果开启了防火墙，需要考虑 NFS 能否透过防火墙提供服务。

3. 创建虚拟磁盘

虚拟磁盘可以由“未使用的物理分区”、“映像文件”、“NFS 共享”来担任。

待模板机制作完之后可能要通过网络将其传输给其他服务器，这时用映像文件来做虚拟磁盘更方便一些。

创建映像文件的命令如下所示：

```
dd if=/dev/zero of=/mnt/xen/vm01.hdimg bs=1M count=20480
```

此命令表达的意思是以/dev/zero 这个特殊设备为模板，将其内容复制到/mnt/xen/vm01.hdimg 设备文件上，bs=1M 的意思是每块的尺寸为1MB，count=20480 是说要分配20480 块，即要创建一个20GB 的虚拟磁盘文件。

4. 开始安装虚拟机

安装虚拟机的命令如下所示：

```
virt-install -n vm01 -b XENbr0 -r 512 -f/mnt/xen/vm01.hdimg --nographics -l/mnt/cdrom
```

具体的参数含义，可以通过执行 virt-install--help 来看一下，如下所示：

- -n 是 domain-U 的名字，配置文件会自动建立到/etc/xen 下。
- -b 用于指定桥接到哪块网卡，“xenbr”是桥接模式的网卡，如果启用的网卡为 eth0，则桥接模式的网卡为 xenbr0，如启用的是网卡 eth1，则桥接模式的网卡为 xenbr1。
- -r 用于指定内存的大小，这里测试时只设置了512MB。
- -f 用于指定磁盘、分区或块设备文件。
- --nographics 表示安装虚拟机的系统时采用的字符界面。如果宿主机没有装 X-window，我们通过字符界面 SSH 连接时还是可以正常显示，所以推荐用字符界面安装。
- -l 用于指定 ISO 安装源，可以是 NFS 路径，也可以是 iso 文件。

此步骤完成以后就进入了虚拟机的控制台。先是启动服务器，然后就到了文本安装界面了。这个安装界面与普通的安装界面唯一不同的地方是，上面的命令中“-l”所指定的 ISO 安装源仅仅起到引导安装的作用。半虚拟化不支持从 ISO 文件中装系统，在真正安装时仍然要配置网络，比如用 NFS。

5. 做一个合格的模板机

如果每次都手动一台一台地安装服务器，效率会极低，而且也很容易出现不符合规范的误操作。所以我们在生产环境中只需要安装一次虚拟机，然后就此虚拟机为模板，克隆出上千台虚拟机用于业务。下面总结一下安装虚拟机模板机要注意的问题：

- 模板机的 IP、主机名要避免现在及以后和线上运行的服务器冲突。
- 模板机应精简不必要的服务，一般只保留 network、SSH、crond 等这类服务。
- 模板机要设置好账号、snmp、nrpe、ssh、免 KEY 登录、文件同步等与监控和安全相关的配置。
- 模板机应安装好如 Apache、Nginx、PHP、Java、Tomcat、Memcached 等常见的服务程序，想用时随时可以开启，省去了编译的时间，统一了编译参数，而且默认未启用时并不会占用系统资源。
- XEN 待配置好模板机以后需要通过网络分发给多人，所以模板机的体积不宜过大，以 20GB 为宜。
- 模板机只有 20GB 的空间，所以要将模板机的分区设定成 LVM，方便动态扩充空间。
- 制作可以让一个模板机修改主机名、IP 地址的脚本，输入 IP 地址和主机名脚本后就可以自动修改。
- 如果是大规模的部署虚拟机，可以考虑搭建内网 yum 源服务器，然后修改模板机的 yum 源为同一内网服务器。

6. 安装完成

在安装过程中可以使用快捷键“Ctrl +]”切回到主机界面中。

在主机界面可以再用“xm console vm01（虚拟机的名字）”切回原来的虚拟机内。

从下面的命令中可以看到虚拟的 Centos4.8 已经安装成功了：

```
[root@ localhost ~]# uname -a
Linux localhost.localdomain 2.6.9-89.ELxenU #1 SMP Mon...
[root@ localhost ~]# cat /etc/redhat-release
CentOS release 4.8 (Final)
```

其中的 XenU 代表虚拟机已经启用了半虚拟化内核。

最后，测试一下虚拟机的网络连通性，如果能连上 SSH，能发布和访问 Apache 搭建的网站，就算是安装完成了（新手要注意，iptables 可能会对上述的网络测试产生影响）。

3.4.4 XEN 寄宿服务器的管理

1. xm 命令

XEN 自带的控制命令是 xm 工具，输入 xm help 后可以看到很多种用法，下面只列出常见的用法。

- xm list：列出所有已知的虚拟机列表。
- xm console：打开控制台管理虚拟机。
- xm create：启动一个基础系统。
- xm shutdown：正确地关掉虚拟机。
- xm destroy：像关掉电源那样关闭虚拟机。
- xm reboot：重新启动虚拟机。
- xm help：所有可用的 xm 命令概述。
- xm top：提供所有虚拟机的状态概貌。
- xm new：添加虚拟机到 XENbase 托管环境。

- `xm start`: 从 XENbase 托管环境启动虚拟机。
- `xm pause`: 暂停虚拟机的活动而不释放使用的内存资源。
- `xm unpause`: 激活使用 `xm pause` 命令暂停的虚拟机。
- `xm save`: 保存虚拟机状态到一个文件。
- `xm info`: 列出 XEN 服务相关的服务器信息。
- `xm restore`: 重新启动已经保存在文件里的虚拟机。

下面就试试这些常见用法:

1) `xm list`: 列出来当前所有虚拟机的名字、ID、分配的内存、CPU、状态和开启时间, 如下所示:

```
[root@ XEN ~]# xm list
```

Name	ID	Mem (MiB)	VCPUs	State	Time (s)
Domain-0	0	745	2	r-----	1401.8
vm01	1	255	1	-b-----	2684.7

其中, Domain-0 的 ID 是 0, 分配了 745MB 的内存, 使用了 2 个 CPU, 当前状态是 r (running), CPU 运算时间是 1401.8 秒。vm01 的 ID 是 1, 它只有 255MB 的内存和一个 CPU, 当前状态是 b (blocked), CPU 运算时间是 2684.7 秒 (这里说的状态指的是硬件相关的状态, 比如说开关机的过程中启动硬件, 比如说执行 `sync` 或 `clock-w` 命令等)。

2) `xm console`: 可以连接到虚拟机的控制台, 后面接虚拟机的名字或 ID, `console` 可以简写。进入控制台后可以按快捷键 “[Ctrl +]” 退出控制台。下面几个命令是相同的:

```
xm console vm01
xm console 1
xm con 1
```

3) 重新开机时输入 `xm list`, 只能看到 domain-0 了。想要看到在启动之前创建的 vm01, 需要使用下面这个命令:

```
[root@ XEN ~]# xm create -f/etc/xen/vm01
Using config file "/etc/xen/vm01".
Started domain vm01
```

其中, `create` 是启动非托管 (默认创建即为非托管) 虚拟机的意思, `-f` 制定的文件正是前面试验中为虚拟机创建的配置文件, 默认的存放路径为 `/etc/xen`。

另外, 下面这个命令也很方便:

```
xm create -f/etc/XEN/vm01
+ =====> xm create -c/etc/xen/vm01
xm console vm01
```

4) 关闭虚拟机的时候, 可以进入虚拟机, 然后执行 `init` 或 `shutdown` 相关的命令。但如果虚拟机已经处于假死状态, 或者想在 XEN 上执行命令, 则可以用如下方法:

```
[root@ xen ~]# xm shutdown vm01
[root@ xen ~]# xm destroy vm01
```

第一个命令向虚拟机系统发出关机指令, 然后由虚拟机系统自动关闭系统。

第二个命令有点暴力但很有效, 模拟拔掉电源的操作。

2. XEN 的套件结构

下面介绍一下 XEN 的套件结构。

- `/usr/sbin/xend`: 该文件是用 python 编写的 xend 的启动脚本。
- `/usr/sbin/xm`: 该命令可以管理 XEN 虚拟机。
- `/etc/xen/xend-config.sxp`: 该文件是设定 XEN 服务本身和 Domain-0 的一些配置文档。
- `/etc/xen/`: 这是 XEN 设定的主目录, 即 XEN 服务自己的设定文档, 它的下级目录是硬件脚本目录和 auto 目录。virt-install 安装后预设的虚拟机配置文件、预设的示例配置文件都在这里。
- `/etc/xen/auto`: 如果你将虚拟机的配置文件放在这里或链接到这里, xend 在启动后就会启动它。
- `/etc/xen/scripts`: 虚拟设备的启动脚本。

3. 虚拟机的配置文件详解

在 `/etc/xen` 目录中, 有虚拟机的示例配置文件 `xmexample1/xmexample2`。文件中对每个具体参数都做了很详细的解释。如果添加一个名为 `vm01` 的虚拟机, 在 `/etc/xen/` 下面就会生成一个同名的配置文件。下面就以此为例, 来了解一下最基本的参数设定。

```
[root@ xen xen]# cat vm01
name = "vm01"
```

虚拟机的名字叫 `vm01`, 也就是 `xm list` 显示的内容。

```
uuid = "92594675-9cb9-f728-b3cf-b252a837033c"
```

定义装置的 uuid 号。

```
maxmem = 512
```

最大内存的设定 (一般和内存设定写成相同数值)。

```
memory = 512
```

内存大小的设定。

```
vcpus = 1
```

CPU 的数量。

```
bootloader = "/usr/bin/pygrub"
```

指定 grub。

```
on_poweroff = "destroy"
on_reboot = "restart"      on_crash = "restart"
disk = [ "tap:aio:/mnt/XEN/vm01.hdisk,xvda,w" ]
```

磁盘设定, 可以增加多块磁盘。

```
vif = [ "mac=00:16:36:18:0d:c9,bridge=XENbr0,script=vif-bridge" ]
```

网络设定。

4. 硬盘的设定

在 XEN 虚拟机中，最“实在”的设备就是硬盘了，所以我们将其单独列出来详细讲解。

在虚拟机配置文件中，“disk = [xxxxxx]”是硬盘信息，详情如下：

```
disk = [ "tap:aio:/mnt/xen/vm01.hdimg,xvda,w" ]
```

其中，“tap:aio: {file}”是指定映像文件为硬盘的意思，部分版本 XEN 中该参数无效，应用“file:”代替。“/mnt/xen/vm01.hdimg”是指定映像文件名的路径；“xvda”表示在该虚拟机内，该硬盘的名称为/dev/xvda；“w”代表有写权限。以上各项用括号括起来表示是一块硬盘的配置文档。如果是多块硬盘，则可以参考如下配置：

```
disk = [ "tap:aio:/mnt/xen/vm01.hdimg,xvda,w","phy:sdc,xvdb,w" ]
```

两块硬盘则用引号划定各自的范围，硬盘之间用逗号进行分隔。“phy: {设备名}”代表这是直接映射的一块物理磁盘、物理分区或逻辑卷；“sdc”表示在 XEN 上该硬件的名字，可以是一块硬盘，可以是一个分区，也可以是一个逻辑卷（lv）。

5. 网络和设定

模板机只用了一块网卡，如果想让虚拟机用多块网卡或不同的网卡来均衡负载，也可以做如下设定。在虚拟机配置文件中，“vif = [xxxxxx]”是网卡信息，详情如下：

```
vif = [ "mac=00:16:36:18:0d:c9,bridge=xenbr0,script=vif-bridge" ]
```

“mac = {0x16}”自然是虚拟出的 MAC 地址了；br0 表示网络连接方式是桥接到 xenbr0，未作特殊设定情况下，eth0 绑定 xenbr0，eth1 绑定 xenbr1……“script = vif-bridge”表示使用桥接脚本，具体的脚本请查看“/etc/xen/scripts”。

如果添加为两块网卡可以用如下方法（最多可以添加为三块网卡）：

```
vif = [ "mac=00:16:36:18:0d:c9,bridge=xenbr0,script=vif-bridge",  
        "mac=00:16:36:18:ad:c1,bridge=xenbr1,script=vif-bridge" ]
```

注：上面的命令实际是一行内容，限于页面宽度断为两行。

3.4.5 XEN 在生产环境下的应用

我们在实际维护 XEN 生产服务器的过程中，摸索出了一套快速部署的方案，同时也逐步意识到了 XEN 虚拟机的不足，下面将分别介绍之。

1. 8 分钟快速部署虚拟机

1) 要有个合格的可以正常启动的模板机，可以是本机安装所得，也可以是从其他服务器拷贝过来的模板级的硬盘文件和配置文件。

2) 将 XEN 服务器上之前保留的空间新建成一个 pv，并加入 vg（卷组），我们假设 vg 名为 vg001。

3) 新建一个 lv（逻辑卷），假设是在 vg001 上新建了一个 60GB 的 www02，命令如下所示：

```
lvcreate -L 60G -n www02 vg001
```

4) 假设模板机的名字为 mould_01，要复制出的服务器为 www_02，请采用下列命令克隆虚拟机。

```
virt-clone -o mould_01 -n www_02 --force -f /dev/vg001/lv001
```


5) 建立软连接到/etc/xen/auto 目录, 让这些业务用虚拟机随实体机一起启动, 如下所示:

```
ln -s /etc/xen/www_02 /etc/xen/auto/www_02
```

6) 模板机分配的 CPU 和内存都很保守, 请根据实际情况编辑虚拟机配置文件中对 CPU 和内存的定义, 命令如下所示:

```
vim /etc/xen/www_02
```

7) 启动虚拟机, 并使用提前写好的脚本快速修改虚拟机的 hostname/hosts/ip 地址等信息, 如下所示:

```
xm cre -c www_02
```

8) 在虚拟机内, 从模板机克隆过来的内容只有 20GB, 剩余的 40GB 可以新建空间, 也可以用 LVM 扩容到现在的 LVM 中。

在操作熟练并适当应用自动化以后, 前两个步骤属于前置准备工作, 不需要花时间; 第 4 步骤克隆虚拟机用的时间最长, 假设模板机内的根分区已用了 5GB 空间, 服务器硬盘可以在 3 分钟内很轻松地完成数据拷贝; 优化过的虚拟机启动很快, 我们给第 7 步骤暂留 2 分钟; 而第 3 步、第 5 步、第 6 步、第 8 步都可以很快完成。结合模板机预先做的设定, 我们有信心“8 分钟提供一个可以马上部署程序的虚拟机”。

2. 虚拟化技术的优点和局限性

了解一个人越深, 越能发现他的优点; 对一个技术掌握得越好, 就越能规避它的缺点。没有完美无瑕的技术, 虚拟化技术也是一样的。我们只有充分了解了虚拟化及 XEN 技术的优点和局限性后, 才能更可靠地将其应用到生产环境中。

(1) 中小规模的应用

在中小企业里, 虚拟化最大的特点是将业务和故障逻辑隔离开。设想一下, 如果一个服务器同时提供 20 个服务会有多混乱, 要是把这些服务分成 20 个虚拟机, 那管理的难度就会大大降低了。加上我们可以做“8 分钟部署”的承诺, 快速部署也成为了虚拟化的优点。但我们要想一下“1 个 OS + 20 个程序”和“1 个 OS + 20 个虚拟机 OS + 20 个程序”的区别, 按照一个虚拟机 OS 占用 200MB 内存来算, 需要为虚拟化花费 4GB 的内存, 硬盘的浪费就更突出了——20 个 OS 本身使用的空间并不多, 只是现在要为每个 OS 预留空间, 那才是最大的浪费。除此之外还要考虑一些软的开销, 比如说网络拓扑变复杂, 比如说公网 IP 地址要付费购买等。

所以说虚拟化节约成本在小规模的应用中根本无从体现, 它最大的好处就是省了运维人员的脑力。

(2) 大规模的应用

大规模应用虚拟化以后, 虚拟机就可以彻底不关心实体机了。如果我们买了 500 台服务器, 然后将其放到虚拟化的“池”里, 并且上面运行了 2000 个虚拟机。这些虚拟机是运行在硬件池中的, 我们不需要关心某个虚拟机是运行在哪个物理服务器上的, 宕机三五台机器甚至不会让硬件池泛起一点波澜。对于天天担心硬件损坏的实体机来说, 虚拟化是一个多么可靠的技术实现啊! 而且因为这个硬件池的存在, 业务的扩充、收缩也变得方便、平滑, 我们可以选择再添加 300 台服务器到这个池子里, 也可以选择从池子里挪出 200 台机器来另作他用, 整个迁移过程可以对虚拟机、对业务做到无痛化。不过, 这么激动人心的技术实现也有其缺点, 那就是钱。VMware 等虚拟化厂

商的售价高到让你瞠目结舌的地步，免费的 XEN 等工具又不具备这么强大的管理功能。如果大规模地应用开源虚拟化技术，公司必然得再招一批系统开发人员对统一管理功能进行大刀阔斧的改造。而且整个周期非常长，投入大，且是否依照 GPL 对成果开源也是个很纠结的商业问题。

(3) 在生产环境对 XEN 的应用

在你没有熟练使用 XEN 之前，请你严格按照前面提到的方法去部署 XEN。那些步骤是我精挑细选的，规避了很多问题，比如说固化内存就可以规避掉一个能导致虚拟机无法上网的 Bug。在保证每个虚拟机都不出故障之后，我们要进一步考虑一个无法回避的问题——I/O 瓶颈。如果你的 RAID 每秒的读写速度是 100M/s，你用的是千兆网卡，但你开了 10 个繁忙的虚拟机，那么每个虚拟机只能忍受 10M/s 的龟速磁盘和百兆网线。即使你为一个虚拟机特地开辟了 500GB 的空间让其存放了上亿个碎文件，但是整个实体机的 I/O wait 始终居高不下。CPU/内存问题我们在克隆虚拟机的时候就会考虑，但 I/O 问题经常被人忽略。那应该如何解决这个问题呢？首先要每次同时上线两台或更多台实体机，同一业务的虚拟机要均匀地分布在多台服务器上，尽力避免出现同一业务的虚拟机集中部署在同一个硬件服务器上的情况，这样出现硬件故障时就不会被“一锅烩”了。每个实体机上部署两个繁忙业务（如热点页面）、三四个不繁忙业务的虚拟机，虚拟机总量不宜超过 6 个。每个实体机的硬盘不大于 500GB，虚拟机是低 I/O 设备，给太大的硬盘它也读不过来，6 个虚拟机每个 60GB 的硬盘差不多了。内存为 16GB ~ 32GB，根据业务给每个机器分 2GB ~ 6GB 的内存足矣。如果真需要给虚拟机特别大的内存，甚至得考虑主板总线 I/O 了。CPU 的核心为 2 × 4 共 8 个，一部分服务器分配双核心，另一部分服务器分配单核心足以应付工作。虚拟机对网络的要求不高，但可以考虑给一些存储量大的虚拟机单独划分网口。

说明 对部分会严重消耗 I/O 的服务器，如 Cache 服务器、数据库服务器，不建议使用虚拟机。

生产环境下 Linux 虚拟化的实践和经验是最宝贵的材料之一，建议大家多关注本节内容。XEN 是开源免费的，网络性能、扩展、稳定性方面是它的强项，现在多将其用于生产环境下的 VPS 中。在企业生产环境中，我们应该在尽可能地熟悉它性能的前提下，扬长避短，最大限度地发挥它的功能。

3.5 Citrix XenServer5.6 虚拟机试用手记

服务器全虚拟化产品 Citrix（思杰）的 XenServer 源自于开放原始码 XEN，和大多数服务器半虚拟化产品相同的是，XenServer 作为一种开放的、功能强大的服务器虚拟化解决方案，可将静态的、复杂的数据中心环境转变成更为动态的、更易于管理的交付中心，从而大大降低数据中心成本。下面，我就以最新的 Citrix XenServer5.6 向大家作演示说明。

这里首先说一下服务器环境，使用 XenServer 要注意以下几个方面，CPU 运算能力要强；内存得足够大；磁盘 I/O 性能要良好。以下为我的 i5 组装服务器的配置，如表 3-2 所示。

表 3-2 XenServer5.6 服务器配置清单

硬件名称	规格型号	硬件名称	规格型号
CPU	Intel i5-760	内存	金士顿 DDR3 1333 2G * 4
网卡	主板自带	硬盘	ST SATA2 1.5T
主板	华硕 P7F-X		

我在 i5 上安装了 6 套不同的系统，有 32 位的也有 64 位的，主要是 Centos 和 Windows 2003。它在公司的机房已稳定运行了 3 个多月，而且测试速度也非常快，所以我也想在自己的家用 Server 上安装测试一下 XenServer5.6。

下面介绍一下完整的安装配置过程。

1. 安装前的准备工作

1) 准备了一块即插即用的 intel 8139 网卡（后来我发现 XenServer5.6 的兼容性很好，这个完全没用上）；

2) 将原先的内存条升级后去掉了一根。我原先用的是 2+1 的条子，后来为了系统稳定，弃用了一根 1GB 的内存条，只用 2GB 的那根。我在安装了 XenServer5.6 后继续安装 VM 时发现，XenServer5.6 对内存的兼容性要求很严格，如果内存条不稳定，安装 VM 时就会报错，导致 VM 安装不成功，这点须切记。

3) 安装 XenServer 时普通 PC 的 CPU 必须支持 64 位及虚拟化的 Intel VT 或 AMD-V 功能，且需要在主板 BIOS 上开启该功能（我找了半天也没找到相关选项，后来发现默认就是开启的）。

4) 准备了一块 80GB 的 IDE 老硬盘（后来也没有用上，XenServer 能识别最新的 SATA2 硬盘）。

5) 家用 Server 机器的配置如下所示：

- ☐ CPU：速龙 64 X2 5000 +
- ☐ 内存：威刚 DDR2 800 2GB
- ☐ 主板：昂达 N61P
- ☐ 网卡：主板自带（intel 8139 弃用）

2. XenServer5.6 的安装及后续工作

XenServer5.6 的安装非常顺利，跟安装 Centos5.5 的过程类似，安装完成后重启即成功了。当然我们的客户机上也要安装 XenCenter，这跟 VMware ESXi 是一样的，不然没办法操控 XenServer5.6。

1) 安装后申请 license。安装结束后的第一件事就是申请及导入一年使用期限的 license 了，这其实很容易，我们可以选中 Tools→license Manager，用它来申请及导入 license 即可免费使用一年了。申请界面也很人性化，注意邮箱地址不要写错，如图 3-9 所示。

成功导入后，XenServer 使用的期限就会变成 2012 年了，如图 3-10 所示。

2) 挂接一个提供 ISO 源的目录，安装 VM 时用网络安装的速度比光盘快多了，这是通过 Linux 下的 Samba 协议来实现的。方法如下：我们选中 Storage→New storage，然后依照图 3-11、图 3-12 和

If the form pre-populated below is not you, click here to delete the cookie and reset the form

Salutation	Select one
First Name	* andrew
Last Name	* you
Organization Name	* cn7788.com
Industry	* Select one
Email	* yuhongchun027@163.com
Business Phone	* 02786911223
Country	* China
Address	* hubei
Address Line 2	
City	* wuhan
州省	* Select one
Zip Postal Code	*
Is the project budget approved?	* Select one
What is your purchase time frame?	* Select one
Are you interested in learning more about the premium XenServer features?	* Select one

图 3-9 申请 license 的网页

图 3-13 来操作。

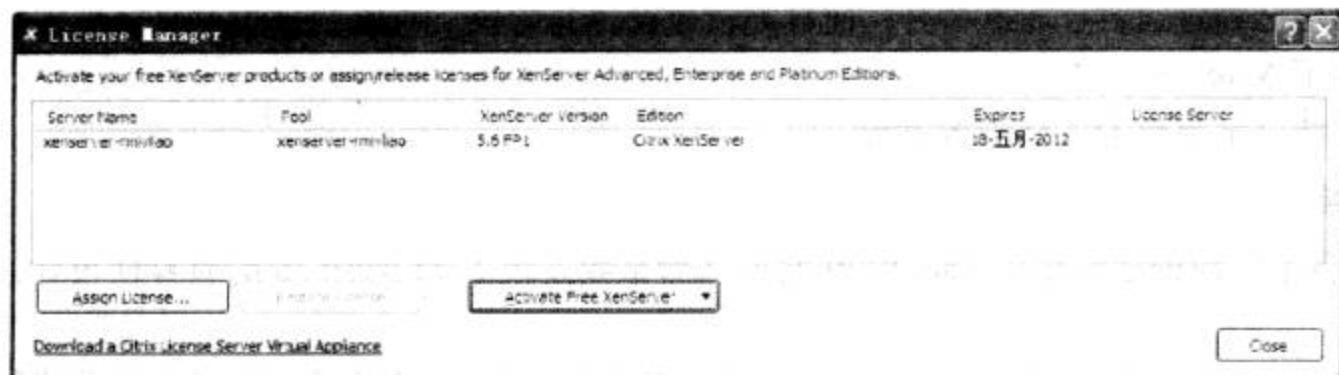


图 3-10 License Manager 功能项显示界面

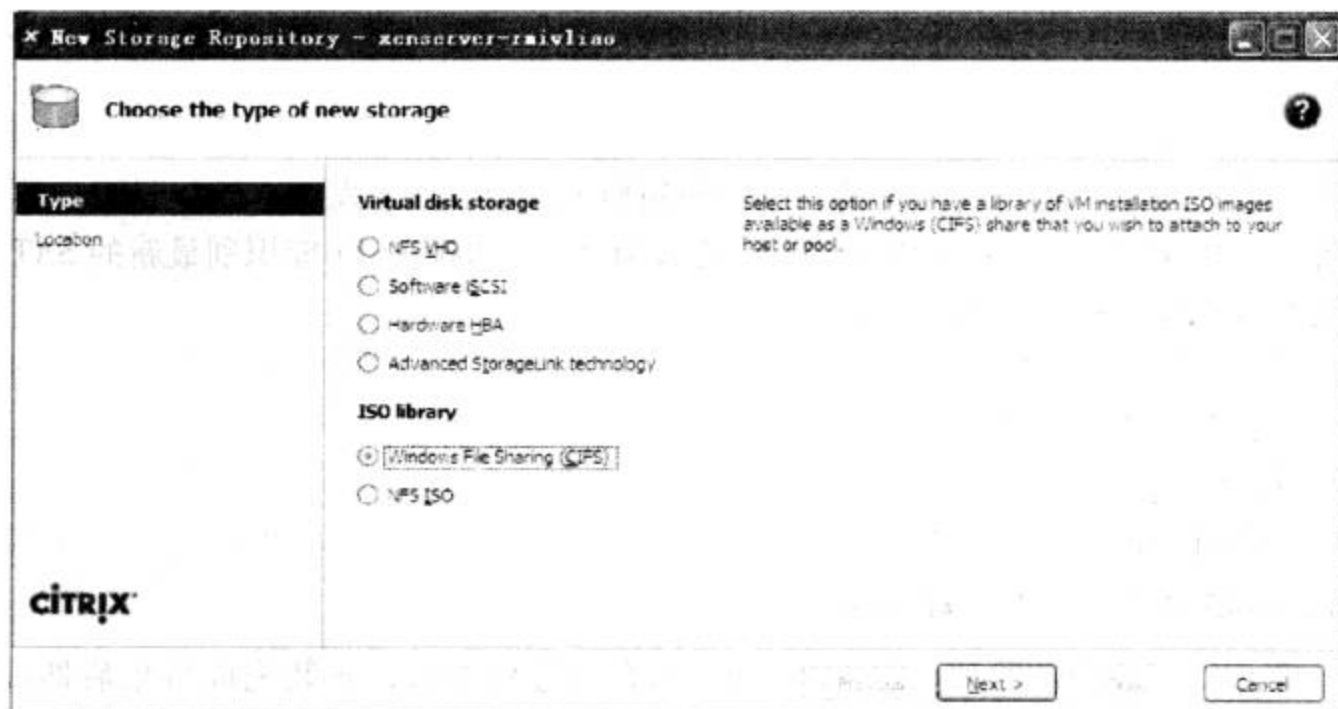


图 3-11 生成一个新的 storage 步骤一



图 3-12 生成一个新的 storage 步骤二

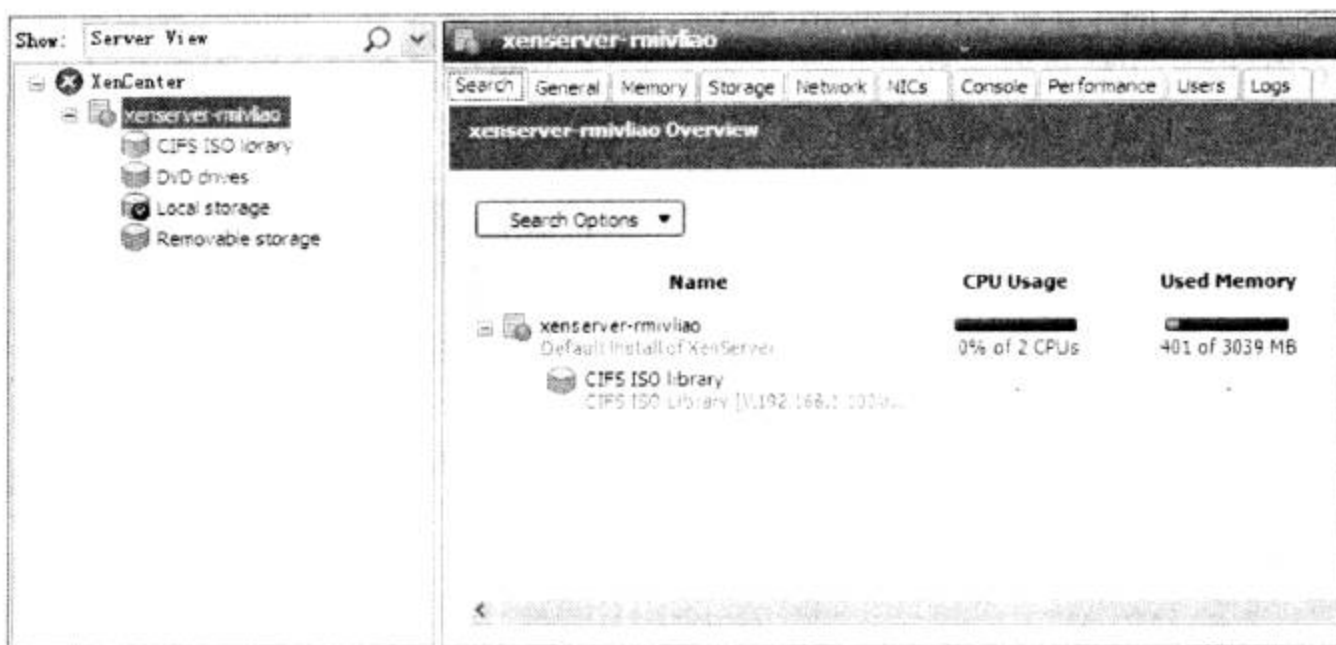


图 3-13 生成一个新的 storage 步骤三

说明 由于选中的 Samba 机器就是我的笔记本，为了方便，我没有配置 administrator 密码。

成功后 XenServer 5.6 会添加一个 CIFS ISO library 的 storage，我们可以把这个当成本地目录来使用。VM 的安装媒介可以选用家用 Server 自带的光驱，也可以用我们新建的 storage。

3) 安装 VM 的方法很简单，跟安装 VMware 的方式几乎一样，因为都是图形化操作，这里也就不详细叙述了。由于 VM 要求每一台机器至少有 512MB 的内存，而我这里只有 2GB 的内存，所以我只安装了两台 CentOS 5.5 x86_64 和一台 Windows 2003 R2 x86_64 的机器，加上笔记本本身就有 4 台机器可以测试使用了，安装后效果图如图 3-14 所示。

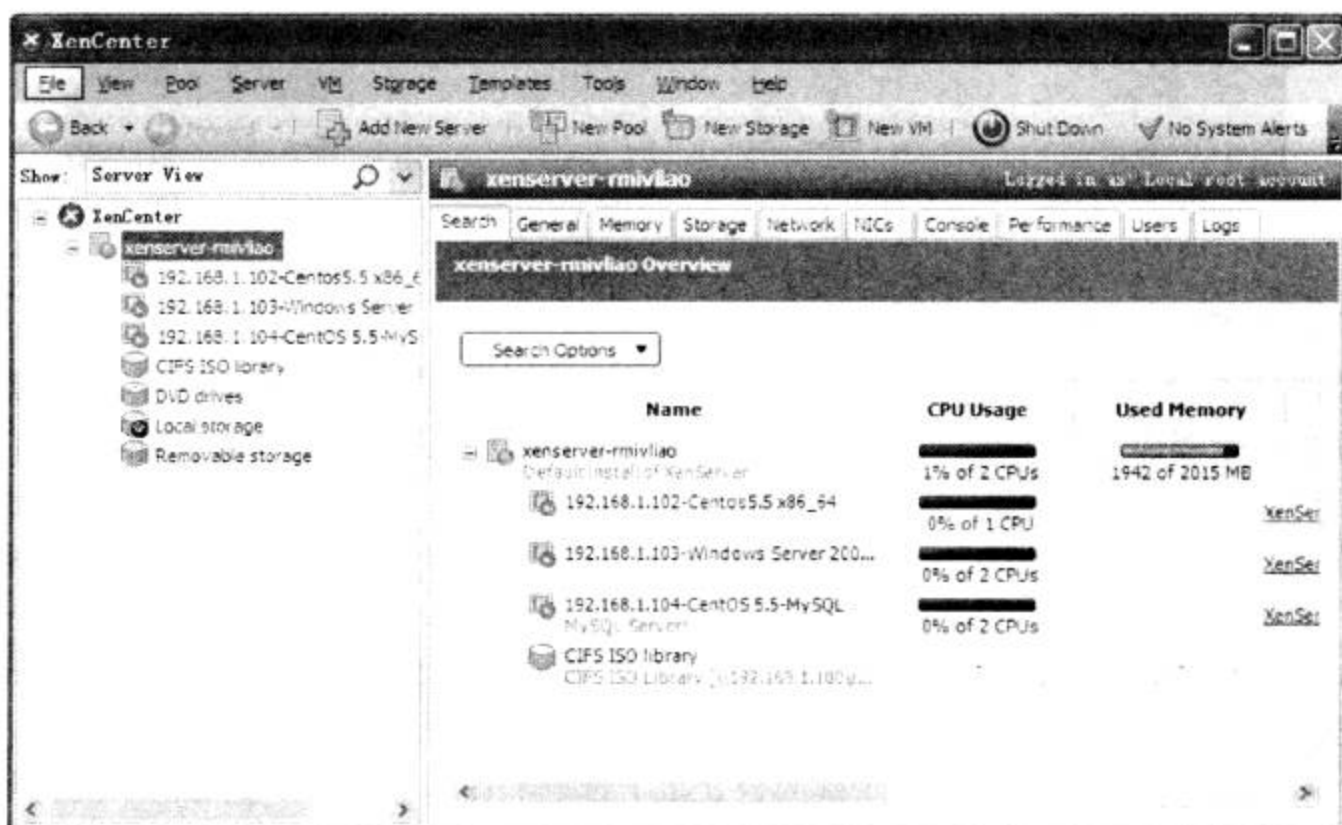


图 3-14 XenServer 5.6 安装后的界面

4) XenServer5.6 的 VM 机器跟 VMware Server 中的 VM 机器一样, 可以设置为自启动模式, 即 XenServer5.6 启动后, VM 机器也随之启动, 如图 3-15 中的配置所示。要实现此功能, 我们选中 “Auto-start on server boot” 选项即可。

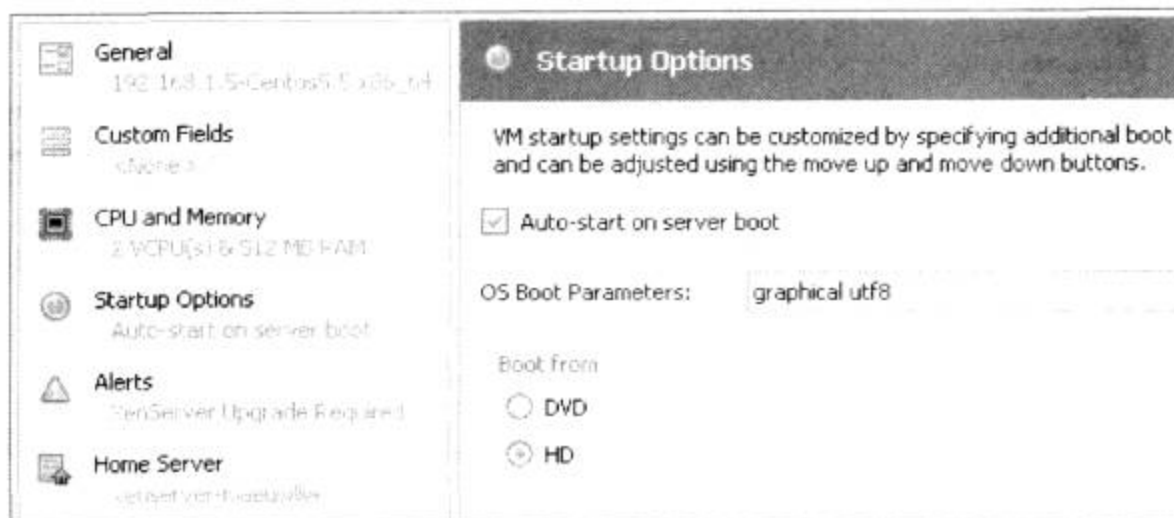


图 3-15 选中 “Auto-start on server boot” 功能项

5) 安装完成后, 我在其中的一台机器上源码安装了 mysql-5.5.12, 并导入了 8.7GB 的测试数据库 project。整个过程中我用 uptime、vmstat 和 iostat 来监测服务器性能, 发现整体性能非常良好, 服务器负载也很小。由于我比较关心磁盘的 I/O, 所以通过下面的一些数据进行了观察, 发现没有硬盘 I/O 瓶颈, 如图 3-16 所示。

avg-cpu:	%user	%nice	%system	%iowait	%steal	%idle						
	0.00	0.00	1.00	0.00	0.00	98.51						
Device:	rrqm/s	wrqm/s	r/s	w/s	rsec/s	wssec/s	avgqu-sz	avgqu-sz	await	svctm	%util	
sda	0.99	0.00	169.31	0.00	23782.18	0.00	140.47	0.22	1.29	0.94	15.84	
sda1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
sda2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
sda3	0.99	0.00	169.31	0.00	23782.18	0.00	140.47	0.22	1.29	0.94	15.84	
dm-0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
tda	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
dm-1	0.00	0.00	169.31	0.00	23528.71	0.00	138.97	0.22	1.29	0.94	15.84	
tdb	0.00	0.00	299.01	0.00	23778.22	0.00	79.52	0.56	1.89	0.60	17.82	
dm-2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
tdc	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	

图 3-16 iostat 监控硬盘性能界面

3. XenServer5.6 进阶使用心得

我已经使用 XenServer5.6 相当长一段时间了, 感觉其稳定性和性能都不错, 所以现在将公司内部机房的虚拟机全部换成了 XenServer5.6 (由于原有的 FreeBSD 的 jail 用得也不错, 所以这部分没有转换)。下面是我的使用心得:

(1) 普通 PC 其实也是支持 XenServer5.6 的, 它的兼容性非常好, 有 64 位架构 CPU 的读者都可以尝试一下看看是否支持, 安装前注意备份硬盘数据。

□ XenServer5.6 支持普通 PC 的板载网卡。以前, 如果要在普通的 PC 上安装 Xenserver5.6, 则需要单独增加一块 PCI 插槽的网卡 (建议使用 Intel、3COM 芯片的网卡), 但 XenServer5.6 的兼容性非常好, 我在昂达、华硕的台式机上都顺利安装成功了。担心网卡问题的朋友可以准备一块 intel 8139 即插即用网卡。

□ 安装 XenServer 普通 PC 的 CPU 必须支持 64 位及虚拟化 Intel VT 或 AMD-V 功能, 且需要在主

板 BIOS 上开启该功能，这一点不多说了，大家应该知道，这是用 Xen 虚拟技术的基础。

- 以前的 XenServer 版本需要老硬盘，新的 XenServer5.6 不怎么挑硬盘，我用最新的 SATA2 硬盘很顺利地就安装成功了。
- XenServer5.6 安装远程控制台 XenCenter 时不需要安装独立的数据库，但需要一台独立的 Windows XP 主机，安装前需要安装 .net Framework 2.0 或以上的版本。我建议大家天空软件下载 .net Framework 3.5 来进行安装。

(2) 如何才能看到宿主机 XenServer5.6 和其 VM 机器的信息？

在 XenServer5.6 的免费图形化管理控制台 XenCenter 上可以看到宿主机及上面运行的所有虚拟机的各种信息和相关图标，例如 CPU 数量、内存大小、磁盘，网卡以及相关的利用率。

(3) 可以用 Linux 的命令来查看 XenServer5.6 的具体信息。

安装了 XenServer5.6 的机器其实就是一台 Linux 机器，我们可以用 `uname` 等命令查看它的具体信息，如下所示：

```
uname -r
```

命令显示结果如下：

```
2.6.32.12-0.7.1.xs5.6.100.307.170586xen
```

用 `lsb_release` 命令查看 XenServer5.6 采用的是哪种版本的 Linux，如下所示：

```
lsb_release -a
```

命令显示结果如下：

```
LSB Version: :core-3.1-ia32:core-3.1-noarch
Distributor ID: XenServer
Description: XenServer release 5.6.100-39215p (xenenterprise)
Release: 5.6.100-39215p
Codename: xenenterprise
```

既然 XenServer5.6 本身就是一台 Linux 机器，我们完全可以用 Linux 查看负载的命令和磁盘 I/O 的命令来监控其性能，比如 `uptime` 和 `top` 及 `vmstat` 和 `iostat`，这些命令包括 Linux 自带的 `free` 和 `df`。当然，只能查看 Xenserver5.6 本身所占用的资源情况，并不能查看其他 VM 机器的相关情况。我们查看一下 XenServer5.6 自身的情况，如下所示：

```
free -m
total      used      free      shared    buffers     cached
Mem:        300        285         14          0         150         32
-/+ buffers/cache:      102        197
Swap:       511          0        511

df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdal        4.0G  1.7G  2.2G  44% /
none            380M   0    380M   0% /dev/shm
/opt/xensource/packages/iso/XenCenter.iso
               44M  44M   0 100% /var/xen/xc-install
//192.168.1.100/ISO  41G  14G  27G  35%
/var/run/sr-mount/efedab53-c882-fe17-f3ed-95f1dd31639c
```

从上可知, XenServer5.6 自身只占用了整个系统 4GB 硬盘和 300MB 的内存, 相当节约资源。

(4) 如果 XenServer5.6 提示找不到硬盘怎么办?

这个问题很好解决, 我们安装系统时并不需要按照 XenServer 5.6 自带的模板来进行安装, XenServer 5.6 的自带模板中没有 Debian 系列, 可以选择 “other install media” 来进行安装, 这样就可以顺利安装 Debian6.0.1a 了。

(5) XenServer5.6 的物理网卡是如何工作的?

在 XenServer 上的物理网卡 (除管理网卡外) 不用配置 IP 信息, 它工作在网桥模式下, 提供了一个通信的通道, 虚拟机的虚拟网卡通过它与外界沟通, 同一个物理网卡负载的虚拟网卡可以被配置成不同的网段、vlan 等, 就如同我们目前的环境一样, 没有任何差别。如果 XenServer5.6 与客户机都是通过防火墙路由上网的话, 那么只需要一块网卡, 推荐 XenServer5.6 使用千兆网卡。

(6) XenServer5.6 怎样使 VM 虚拟机达到比较高的 I/O 性能?

XenServer5.6 下的 Linux 的 VM 虚拟机会直接使用 Linux 的驱动, 效率接近传统的 PC 方式。安装 Windows 的 VM 虚拟机之后需要安装 xen tools 来替换原来的模拟驱动, 可大大提高 I/O 效率。另外, 如果一个物理服务器上运行着多个虚拟机, 则建议使用存储, 可以大幅度提高存储效率。需要强调的是, XenServer5.6 提供了新功能 Storagelink 帮助虚拟机直接使用存储的高级功能, 这样可以大大提高 I/O 性能。虚拟机如果不带存储, 只能算虚拟机而不能算是虚拟化, 如果大家有存储的话尽量搭配着 XenServer5.6 来使用。

(7) 如何规划 VM 虚拟机? 有没有量化标准?

这个完全取决于虚拟机的配置和负载压力, 一个简单的配置方式就是按照内存来分配, 各虚拟机内存分配量 + XenServer 使用内存 = 物理服务器内存。在家用的 XenServer5.6 上我装了两台 200GB 硬盘、512MB 内存的 Centos5.5 x86_64 虚拟机, 平时主要用来调试 SHELL 和 Python 脚本, 再就是做 Puppet 和 Rsync + Inotify 等实验, 有时也做一下 MySQL 的主从复制等。另外这两台机器也用作 vsftpd 的备份机器, 一台是文档 + 软件的备份, 一台是影视剧的备份, 比如在里面收藏了一套中英文字幕的《老友记》, 用来锻炼英文口语, 感觉非常方便。

3.6 小结

本章分别介绍了各种不同的虚拟化技术, 有 Windows Server 2003 下的 VMware GSX Server 和 VMware Server, 还有 Centos5.5 自带的 Xen 及思杰的 XenServer 5.6。大家可以根据实际环境部署应用。如果是生产环境下的虚拟化实施, 可以参考曹亚孟的 Xen 实施文档。可为企业节省大量采购资金和人力成本, 这是 Linux 虚拟化最有价值的优点之一。



第 4 章

生产环境下服务器的故障诊断与排除

- 4.1 快速排障的重要性和必要性
- 4.2 安装系统时容易发生的错误描述与处理方法
- 4.3 网络配置时容易发生的错误描述与处理方法
- 4.4 系统维护时的注意事项
- 4.5 紧急处理线上服务器故障的办法
- 4.6 检查机房应注意的位置和细节问题
- 4.7 系统维护时应注意的非技术因素
- 4.8 小结

服务器系统与平时用的办公系统或家用系统不一样，它要求能 365×24 小时不间断地工作，以便为我们提供服务。这也是我们不能随便重启服务或服务器的原因。试想，如果我们上网正欢时所用的 DNS 服务器重启了，那我们该是何等郁闷啊！有些朋友喜欢在 Windows XP 下用 Ghost 软件来重新恢复已经严重崩溃的系统，不过此举对于服务器而言却没有意义。我们在系统发生故障时会尝试修复它，而不是重装。请大家记住服务器系统与一般系统的这一区别。

4.1 快速排障的重要性的必要性

我们的服务器多是线上的生产服务器，基本都是负责电子商务或 CDN 节点的，如果发生故障，导致大量单据丢失，就会严重影响客户的信心，继而影响到公司的信誉。所以，在服务器发生故障时（此时我们的手机会收到 Nagios 的短信报警），我们一定要迅速排除故障，在最大程度上降低公司的损失，这也是系统管理员的职责所在。

4.2 安装系统时容易发生的错误描述与处理方法

如果购买了新服务器，我们最先做的事情肯定是检测硬件和安装服务器系统。不过，在安装过程或调试过程中有可能会出现问题，我们应该怎样正确地处理它们呢？本节将集中讨论如何处理这一类问题，希望能帮助大家进一步了解系统相关知识。

4.2.1 忘记了 Centos5.5 的 root 密码怎么办

我的线上 Linux 服务器一般会设置 28 位密码，就算是内网开发的机器，所设置的密码也比较长，有时候就会出现忘记 root 密码的情况。忘了 Centos5.5 的 root 密码怎么办？别着急，按下面的流程一步步走的话，会很容易解决问题。

1) 在开机启动的时候按空格键能看到 Centos5.5 目录（3 秒后将进入默认设置中），用上下键选中你要进入的那个内核，选中后按键盘上的“E”键，进入如图 4-1 所示的界面。虽然图中只有一个 Centos5.5 引导，不过如果你升级了系统或安装了 XEN 虚拟化后，就会有多个显示了。

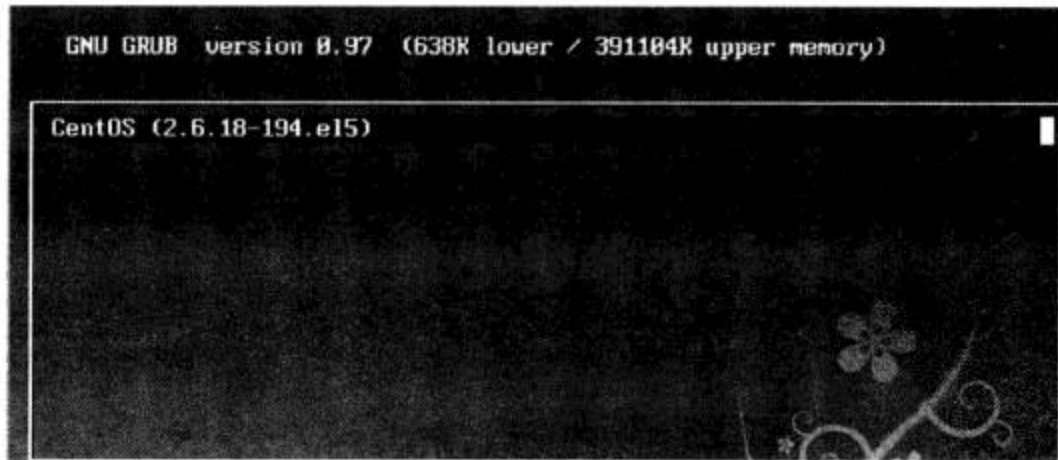


图 4-1 Centos5.5 的引导界面

2) 看到图 4-1 所示画面后，继续按“E”键选择内核，进入如图 4-2 所示的画面。

3) 在图 4-3 所示的画面可以编辑，在信息的最后加空格，然后键入“single”或“S”，或者直接输入数字“1”并回车，确定进入下一步（均可以进入单用户模式）。

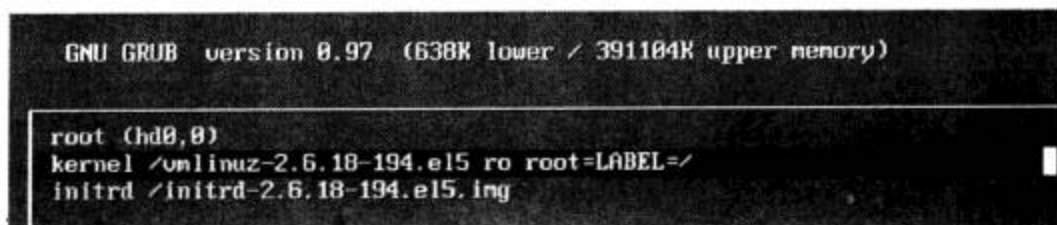


图 4-2 Centos5.5 的 GRUB 界面

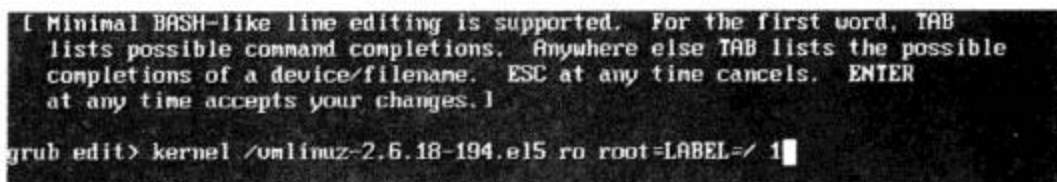


图 4-3 Centos5.5 的 Kernel 编辑界面

4) 进入到如图 4-4 所示界面，然后可以按“B”键启动系统。

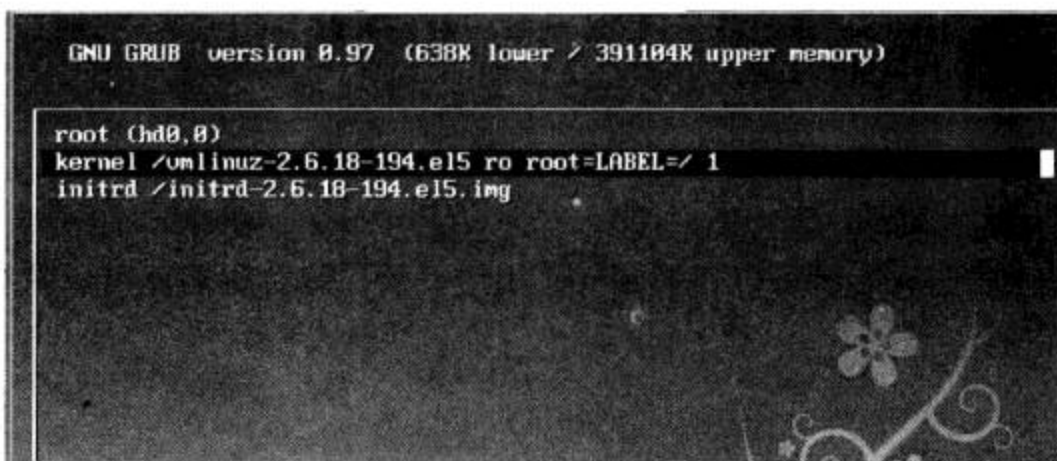


图 4-4 Centos5.5 的 GRUB 界面

5) 进入如图 4-5 所示的单用户界面后，在这个画面中的“#”后输入“passwd root”，重新设置 root 的密码，然后再确认输入一遍，即重设了 root 密码。进入 Centos5.5 单用户模式后，我们可以输入 reboot 重启计算机，此时的 root 密码就被更新了，如图 4-6 所示。

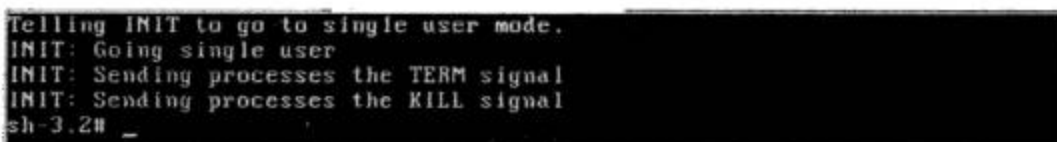


图 4-5 Centos5.5 的单用户模式

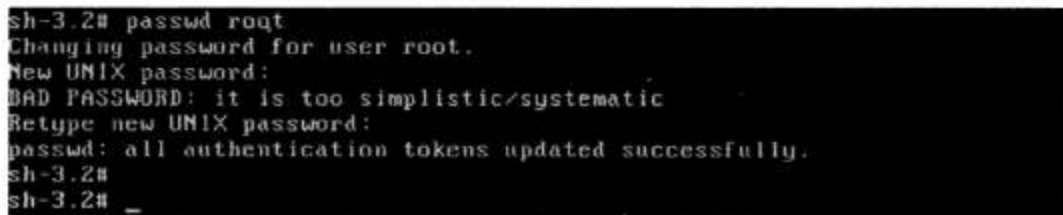


图 4-6 root 密码显示已更新

另外，需要说明一下，如果我们维护的系统是 FreeBSD8.1 的话，很多时候系统管理员们可能会忘记 root 密码，但由于大家都已经对在其下用普通用户进行 sudo 操作很熟悉了，再加上

FreeBSD8.1 默认是不允许我们用 root 用户远程登录的，此时若重新设置 root 密码就显得有些多余了。所以我们在维护 Centos5.5 系统时可以借鉴 FreeBSD8.1 的做法，配置了 root 密码以后，立即配置一个具有 sudo 权限的用户 andrewy，步骤如下。

1) 在 Centos5.5 下新建一个系统管理用户 andrewy，这里以 root 用户来操作，如下所示：

```
useradd andrewy
passwd andrewy
```

2) 将 andrewy 赋予 sudo 权限，即 visudo，添加如下代码：

```
andrewy ALL = (ALL) ALL
```

3) 如果忘记了 root 密码，我们可以以 andrewy 的身份进去，然后切换到 root 身份下修改密码，如下所示：

```
[andrewy@ research ~] $sudo su -
We trust you have received the usual lecture from the local System
Administrator.It usually boils down to these three things:
#1)Respect the privacy of others.
#2)Think before you type.
#3)With great power comes great responsibility.
[sudo] password for andrewy:
```

我们输入 andrewy 的密码后就可以切换到 root 用户下面，然后我们就可以执行 passwd 命令修改 root 密码了，如下所示：

```
[root@ research ~]#passwd
Changing password for user root.
New UNIX password:
BAD PASSWORD:it is based on a dictionary word
Retype new UNIX password:
passwd:all authentication tokens updated successfully.
```

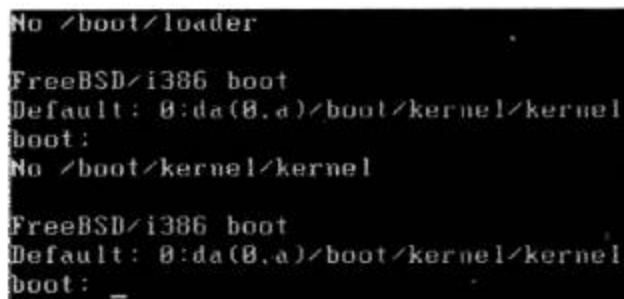
相对而言，我更推荐采用第二种方法来确保 root 密码不会被遗忘。无论是安装 Centos5.5 还是 FreeBSD8.1，当我们无法去机房操作时，就需要一个具有 sudo 权限的用户了。当然，这个用户的密码也需要符合复杂密码要求，例如像 p@sSw0rd_192.168.1.102_nagios 这样字符组合的密码，被破解的几率还是很低的。

4.2.2 正确重设 root 密码

有时候因为工作的需要，我们需要重新配置 root 密码，这个时候我们肯定会输入一个比较长而且复杂的新密码。有些比较性急的管理员在输入新密码后就退出当前终端，回公司工作了，回到公司却突然发现，新 root 密码进入不了系统。虽然能够回机房重设 root 密码，但这样严重影响了工作效率，怎么办？其实这里面也有个小技巧，推荐给大家。我们在当前终端下重设了 root 的新密码后，不要急着退出，这时候我们可以用 [Alt + F2 ~F6] 键进入另一终端，尝试使用 root 的新密码登录。如果确认无误，再退出所有终端，这样失误的几率就非常低了。我个人还是比较推荐在机器上保留一个具有 sudo 权限的用户。当然了，有时因为安全要求，服务器上除了 root 以外不能保留其他具有 sudo 权限的用户，这时候更改 root 密码就要慎之又慎了。

4.2.3 安装 FreeBSD8.1 时不要设置/boot 分区

用惯了 Linux 的朋友都知道 Linux 下有 /boot 分区，我第一次安装 FreeBSD8.1 时也跟安装 Linux 一样分配了 /boot 分区，结果机器启动不了，需要手动引导，相信很多人都遇到过类似的情况。分配了 /boot 分区后，在系统安装完成重启后便出现如图 4-7 所示的故障。



```
No /boot/loader
FreeBSD/i386 boot
Default: 0:da(0,a)/boot/kernel/kernel
boot:
No /boot/kernel/kernel
FreeBSD/i386 boot
Default: 0:da(0,a)/boot/kernel/kernel
boot: -
```

后来通过在网上查资料，了解了 FreeBSD8.1 手动引导的全过程，下面简单描述一下。由于独立分区 /boot 造成了 FreeBSD8.1 在引导过程中无法正确找到内核引导的位置，所以才会产生上述问题。通过以下命令可以解决此问题，如下所示：

```
boot:0:da(0,c)/loader
```

图 4-7 FreeBSD 分配了 /boot 分区进不了系统故障图示

注意 这个引导盘符根据 da0s1x 的 x 得来，因此 FreeBSD8.1 系统的 /boot 所在的分区区号才是真正的 x 字母，如果不知道就从 x 字母往后试试。

解决了引导问题后，进入 loader 界面。由于默认的 kernel 位置是 /boot/kernel，所以依然需要手动加载，输入如下命令：

```
ok load kernel/kernel
```

获得 kernel 信息后，输入如下命令：

```
ok boot
```

这样就可以正常引导了。

但是随后还需要在磁盘挂载的时候输入以下命令：

```
mount root>ufs:/dev/da0s1a
```

然后便进入系统了。可是，不能每次重启都手动启动，太麻烦了，所以这里强烈建议大家安装 FreeBSD8.1 时不要分配 /boot 分区。我在安装 FreeBSD8.1 时，会严格遵守工作流程来操作，即只分 3 个区：根分区、/data 分区及 swap 分区，这样出错的几率会小很多。

4.2.4 Centos5.5 的 Grub 引导程序出错

修改了 Centos5.5 的 grub.conf 后，或者硬盘的物理位置和分区发生变化后，将造成系统不能正常启动。这时可进入 rescue 模式，修改 grub.conf。

一般，Grub 出错的情况如下：

□ 如果没有找到内核（vmlinuz-x.x.x 文件），就会出现如下显示：

```
File not found
Press any key to continue....
```

说明 root (hdx, x) 错误，或内核文件名不对。

□ 如果找到内核后，运行一会儿出现如下显示：

```
Kernel Panic:Not init Found
```

一般来说是没有找到根分区（/分区），即 `root = /dev/xxx` 不对。

□ 如果找到内核后，运行一会儿出现如下显示：

```
Kernel panic:VFS:Unable to mount root fs on ...
```

一般来说可能是忘了加上 `initrd/initrd-2.4.21-4.EL.img`（大多发生在使用 scsi 硬盘时）。

总的来说，`grub.conf` 里面必须存在如下几行内容。我们可以用 `vim` 命令来修改 `/boot/grub/grub.conf` 文件的内容，如下所示：

```
title linux
root (hd0,0)
kernel/vmlinuz-2.6.18-53.el5 ro root = LABEL = /
initrd/initrd-2.6.18-53.EL.img
```

`root(hd0, 0)` 表示 `/boot` 分区所在位置。

`kernel/vmlinuz-2.6.18-53.el5 ro root = LABEL = /` 表示此项为内核和根分区（/分区）位置（根分区可能是 LVM 和 RAID，而不仅是 `hdx` 和 `sdx`，这要看实际情况。有个命令我经常用：`findfs LABEL = /`）。

如果我们平时想熟悉和练习快速排障能力，可以在测试机上删掉 `grub.conf` 文件，每次重启时自己使用 Grub 的交互命令行操作就会熟悉文件里面的内容了。

4.2.5 安装 Centos5.5 时忘了关闭 iptables 和 SELinux

有些朋友安装 Centos5.5 时，忘了关闭 `iptables` 和 `SELinux` 选项，结果导致系统出现许多莫名其妙的问题，特别是在进行一些流量转发（例如 `LVS`、`Nginx`）时。所以我们在进入系统后，可先用命令关闭它们，然后根据实际需求决定是否打开它们，命令如下：

```
system-config-securitylevel
```

输入此命令后，就会进入防火墙配置界面，如图 4-8 所示：

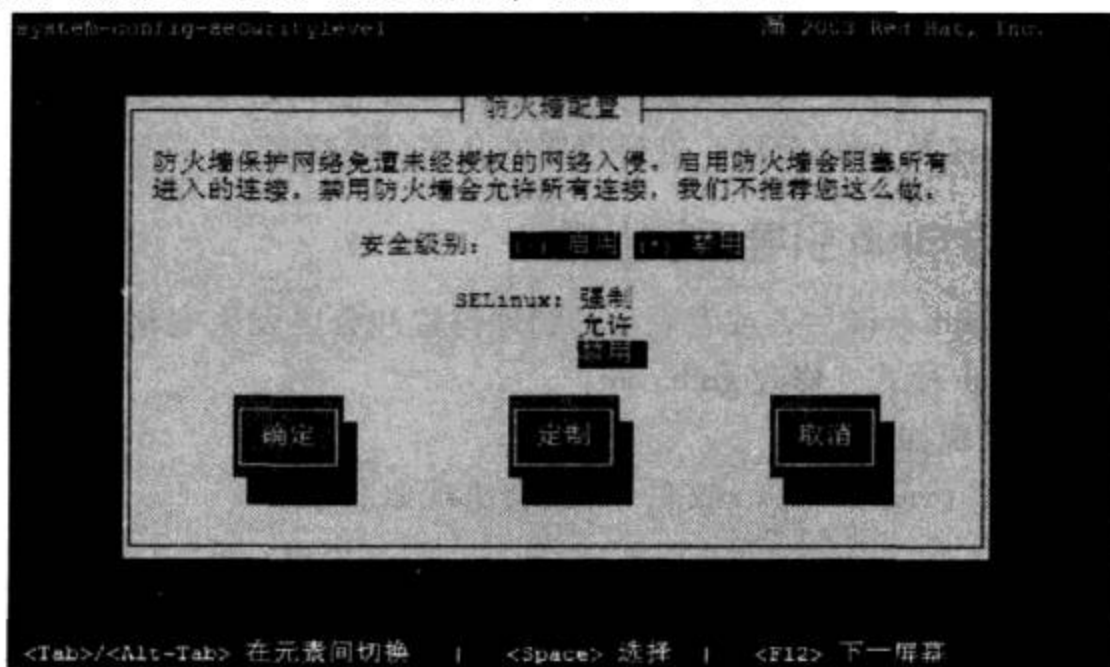


图 4-8 Centos5.5 防火墙配置界面

配置完成后我们可以用命令查看一下，正确关闭 iptables 和 SELinux 时应显示如下内容：

```
[root@localhost ~]# iptables -nv -L
Chain INPUT (policy ACCEPT 279 packets, 21732 bytes)
pkts bytes target    prot opt in     out    source        destination
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target    prot opt in     out    source        destination
Chain OUTPUT (policy ACCEPT 756 packets, 60241 bytes)
pkts bytes target    prot opt in     out    source        destination
[root@localhost ~]# getenforce
Disabled
```

上面是图形化的操作，推荐使用。如果是用命令行的方法操作，也会比较方便。关闭 iptables 的命令如下所示：

```
service iptables stop
chkconfig iptables off
```

要关闭 SELinux，可使用以下命令行临时关闭之。SELinux 由开启状态转为关闭状态后，最好是重启一下服务器。

```
setenforce 0
```

4.2.6 如何解决 Putty 或 PieTTY 的乱码问题

解决 Putty 或 PieTTY 的乱码问题的方法如下：先打开 Putty 主程序，依次选择 window→Appearance→Font settings→Change，然后选择 Fixedsys 字体，字符集选择 CHINESE_GB2312。在 window→Appearance→Translation 的 Received data assumed to be in which character set 中，把 Use font encoding 改为 UTF-8。一般这样就可以解决问题了。如果还没有解决，咱们接着往下看。

即使我们在安装 Linux 的时候选择的语言是中文，在用 Putty 或 PieTTY 远程管理的时候，可能看见的中文文件夹还是乱码显示的。以下方法可以帮我们解决乱码问题：

□ console 终端乱码。在 /etc/profile 文件的最后一行添加如下内容：

```
export LC_ALL="zh_CN.GB18030"
```

□ Xwindow 终端乱码。在 /etc/sysconfig/i18n 文件的最后一行添加如下内容：

```
export LC_ALL="zh_CN.GB18030"
```

其实 Putty 出现乱码不外乎就是编码及字符集的原因，我在 RHEL4、RHEL5 及 Centos5.2 均能通过以上步骤解决问题。要是还不行干脆就放弃 Putty，改用 Xmanager3 企业版的 xshell3.0。xshell3.0 目前是 Linux 运维工作人员的首选工具，而且遇到乱码问题仅仅需要采用第二步就能解决。

4.2.7 安装双系统时不小心删除了 Grub 所在的分区

由于工作需要，我出差在外时要在自己的工作机上安装 Windows XP SP3 和 Centos5.3 系统，可我一不小心删除了 Grub 所在的分区/dev/hdb8，搞得连 Windows XP 也进不了了，而这已经是我第二次犯错。由于我的工作机上没有光驱和软驱（以前本着经济的原则配置的），所以上次借别人的

光驱才恢复。我这次想靠自己来解决这个问题。能不能尝试 U 盘启动呢？我花了不少时间把一台电酷闪 8GB 的 U 盘配成了 USB - CDROM + USB - HDD 双启动的 U 盘，工作机居然不支持，怎么办？这时忽然想起工作站是支持网络引导的，那就应该有办法了，所以我用上了网刻软件 MaxDOS_71PXE_G115.exe，如图 4-9 所示。

搞定后原以为万事大吉了，可我发现一启动还是回到了 Grub 报错的状态。我决定不选择“克隆结束后重启客户机”，这样就能回到 DOS 界面下了。然后选择一款 MBR 修复软件 Diskgen 或 spfdisk；或者直接使用命令 fdisk/mbr。其实还有个办法：在“grub>”提示符后输入：“rotnoverify (hd0, 0)”，回车将第一块硬盘 (hd0) 的第一个分区 (0) 设为根分区/root 设备，但不加载文件系统；在“grub>”提示符后输入：“chainloader +1”，然后回车，将启动引导权转交给当前分区的首扇区（Windows XP 系统所在分区的首扇区），这样也能解决问题。

另外，为了加深大家对 Linux 系统启动的理解，这里附上我以前整理的 Linux 系统（含 RHEL 及 Centos 系列，这里以 Centos5.5 为例说明）启动顺序，建议大家将其记住，因为现在许多企业在面试时都会问到这个问题。Linux 启动顺序如下：

(1) BIOS 初始化。

(2) MBR（大家可以注意一下，Centos5.5 的 bootloader + 分区表 + 结束位 = 512B）。

(3) GRUB。

□ 加载内核 Kernel 为只读；

□ 加载 RAMDISK。

(4) INIT 进程，运行的文件如下：

1) /etc/inittab：此文件在 Centos5.5 中属于重要文件之一，它用于设定系统预设的 runlevel 级别，并指定终端机启动的个数，设定 Ctrl + Alt + Del 组合键的动作等。

2) /etc/rc.d/rc.sysinit：此文件的作用如下。

□ 启动 Udev（热插拔设备）并且启用 SELinux。

□ 把 Kernel 参数设定在/etc/rc.sysctl.conf 中。

□ 设定系统时间。

□ 载入 Keymaps 设定（注：Keymaps 定义键盘）。

□ 启用 swap 交换分区。

□ 设置定义主机名，主机名路径/etc/sysconfig/network。

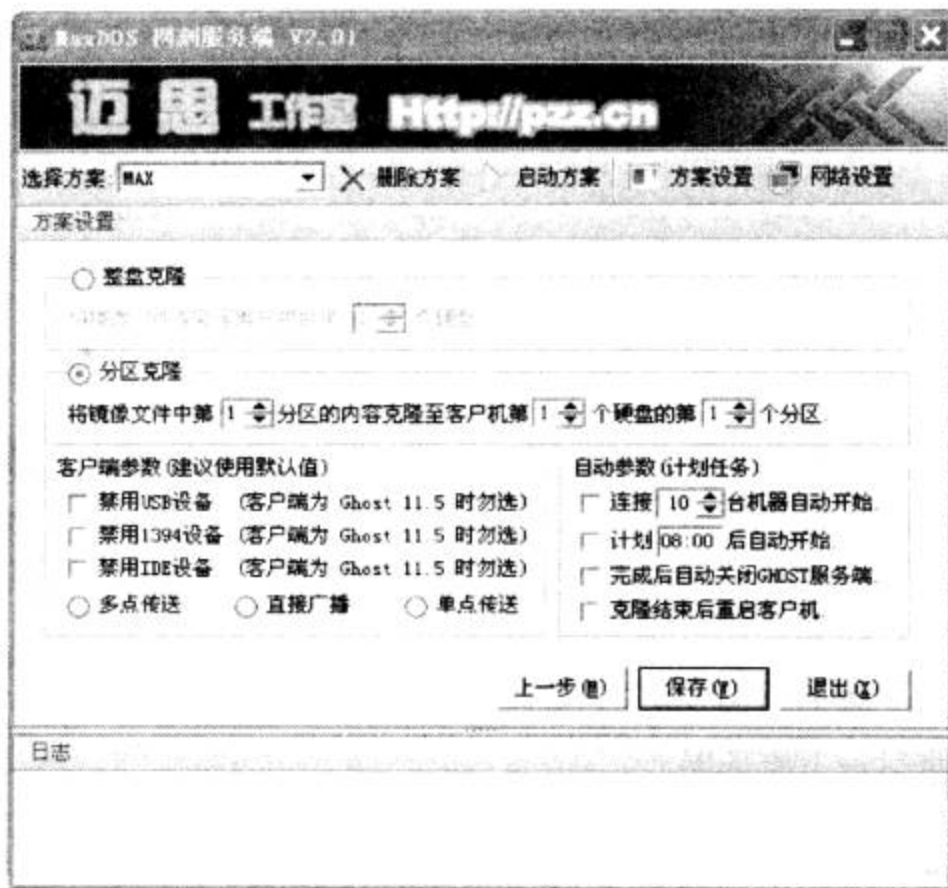


图 4-9 MaxDOS_71PXE_G115 工作界面

- ☐ 检查根目录，并且重新挂载可读可写的状态。
- ☐ 启用 RAID 和 LVM 设备。
- ☐ 启用 Disk quotas 功能，可以限制用户最多能用多少空间。
- ☐ 检查 Linux 的其他目录，并且把它们重新挂载。
- ☐ 清除一些开机时的没用文件。

3) /etc/rc.d/rc3.d/：目录下的文件全部执行，如果是在 5 模式下的话，则执行/etc/rc.d/rc5.d/ 目录下的所有脚本文件。

4) /etc/rc.d/rc.local：此文件最后一个执行，相当于 MS 中的 autoexec.bat，我们也习惯将一些自启动脚本置于此文件中。

(5) Login。

4.3 网络配置时容易发生的错误描述与处理方法

系统成功安装后，下一步就是正确配置服务器的网络情况了。在此过程中，如果有人对 Linux 系统或 BSD 系统不太熟悉，也很容易犯一些错误，比如说网卡配置有问题导致 IP 上不去、DNS 配置失误等。

4.3.1 安装 Centos5.5 时忘了激活网卡

许多朋友刚开始安装 Centos5.5 时，忘了勾选“Active on Boot”，最直接的后果就是导致网卡 IP 分配不成功，下面是正确的步骤，如图 4-10 所示。

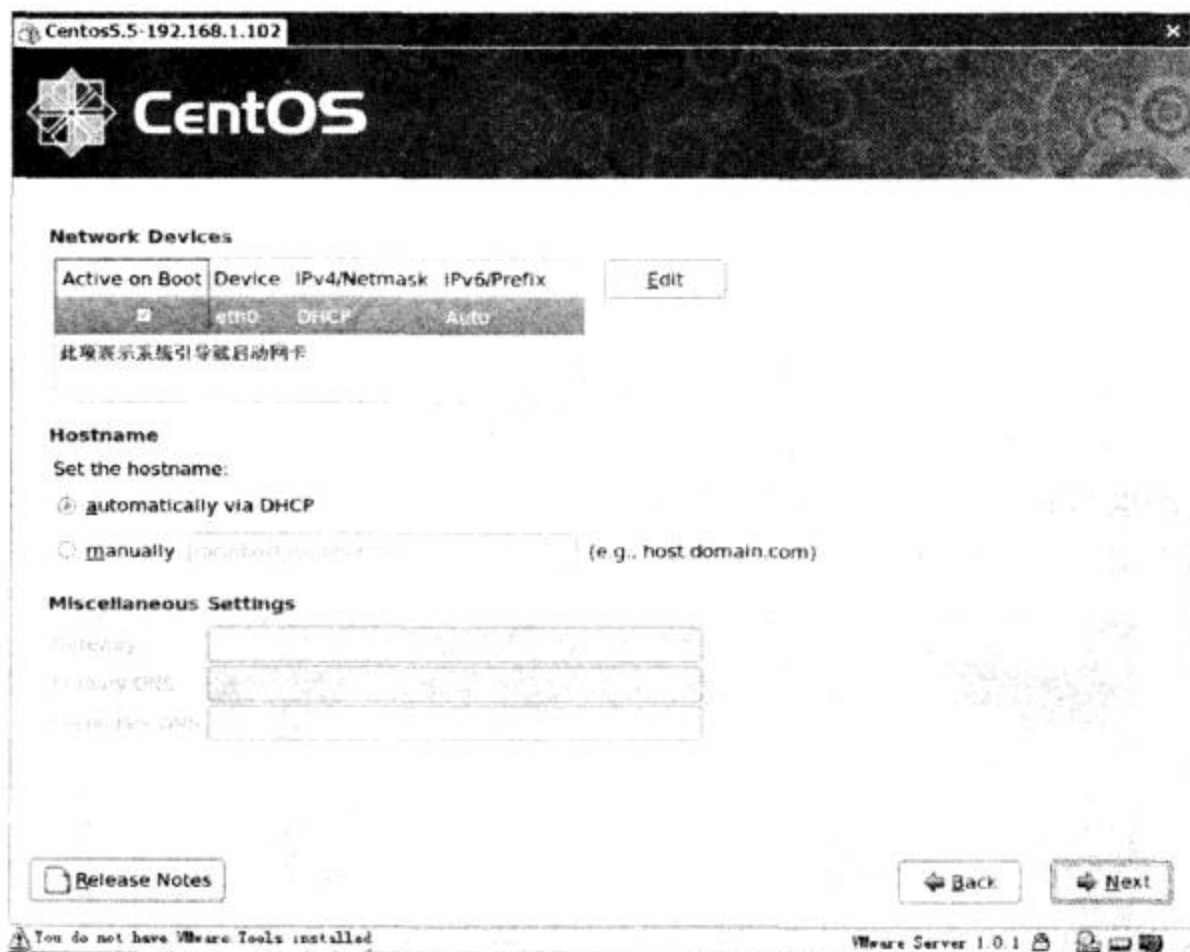


图 4-10 Centos5.5 配置网卡界面

我们进入系统后会发现，IP 地址没有正确分配。不过也不要着急，我们可以直接配置 eth0 网卡文件，即 /etc/sysconfig/network-scripts/ifcfg-eth0，文件内容如下：

```
DEVICE=eth0
BOOTPROTO=None
HWADDR=00:14:22:1B:70:FA
ONBOOT=yes #这项必须选择 yes,表示系统启动时就引导网卡
NETMASK=255.255.255.192
IPADDR=203.93.236.145
GATEWAY=203.93.236.129
TYPE=Ethernet
```

在上面我们将 ONBOOT=no 改成 yes，然后重启网卡服务即可以解决这个问题，命令如下：

```
/etc/init.d/network restart
```

4.3.2 Centos5.5 网卡文件备份的正确方法

许多朋友备份 eth0 网卡文件时喜欢跟备份其他文件一样，用命令 `cp /etc/sysconfig/network-scripts/ifcfg-eth0 /etc/sysconfig/network-scripts/ifcfg-eth0.bak`，然后用 `/etc/init.d/network start` 来启动机器的网卡。这其实会带来一个问题：系统也会启动以 .bak 为后缀的网卡文件，请大家注意 Centos5.5 的 /etc/init.d/network 脚本的下面这一段内容：

```
# find all the interfaces besides loopback.
# ignore aliases, alternative configurations, and editor backup files
interfaces=$(ls ifcfg* | \
    LANG=C sed -e "$__sed_discard_ignored_files" \
        -e '/\.(ifcfg-lo\|:\|ifcfg-.* -range\)/d' \
        -e '/ifcfg-[A-Za-z0-9\._-]+\$/ { s/^ifcfg-//g;s/[0-9]/ &/ }' | \
    LANG=C sort -k 1,1 -k 2n | \
    LANG=C sed 's///')

rc=0
```

系统会将此目录下以 ifcfg 开头的所有网卡均启动，大家看到这里应该会明白，此时的备份方式其实是有问题的，如果用 Centos5.5 下自带的工具 setup 进行配置，从图 4-11 中可以明显看出，系统将会把 bak 文件也当做一块网卡。如果是搭建测试环境的话，很有可能影响实验结果，所以正确的备份应该重新定义一个不以 ifcfg 开头的名字。

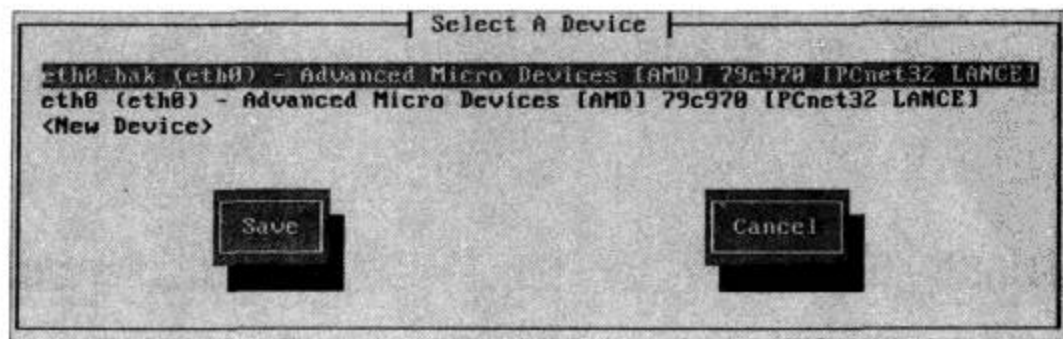


图 4-11 出现 eth0.bak 网卡的错误界面

4.3.3 解决远程桌面超出最大连接数的问题

有时管理一台 Windows 2003 服务器，远程连接时账号、密码都输入完了，点击连接后却弹出一个“终端服务器超出最大允许连接”的提示。上网查询并归纳了一下出现这种情况的原因和解决办法。

1. 原因

用远程桌面连接登录到终端服务器时经常会遇到“终端服务器超出最大允许连接数”或类似的错误，导致无法正常登录终端服务器。引起该问题的原因在于终端服务器默认的连接数为2个连接，如果登录远程桌面后没有采用注销方式退出，而是直接关闭了远程桌面窗口，那么实际上 Session 并没有释放掉，而是继续保留在服务器端，这样就会占用总的连接数。当这个数量达到最大允许值时就会出现上面的提示。

2. 如何解决这个问题

1) 用注销 (log off) 来退出远程桌面而不是直接关闭窗口。

2) 限制已断开连接的 session 存在时间。

3) 增加连接数。

如果此时我们需要紧急进入系统工作，可以采用命令行的方式进入，方法如下：

依次选择开始→运行→mstsc/console/v: 服务器名、IP 或域名。

例如：

```
mstsc/console/v:8.8.8.8
```

或者

```
mstsc/console/v:www.cn778899.com
```

mstsc.exe 的语法如下：

```
mstsc.exe {ConnectionFile | /v:ServerName[:Port]} [/console] [/f] [/w:Width/h:Height]
```

其参数说明如下：

ConnectionFile 是指定 mstsc.etx 用于连接的 .rdp 文件的名称。

下面的选项分别为：

☐ /v:ServerName[:Port]

指定要连接的远程计算机和（可选）端口号。

☐ /console

连接到指定的 Windows Server 2003 家族操作系统的控制台会话上。

☐ /f

在全屏模式下启动“远程桌面”连接。

☐ /w: Width/h: Height

指定“远程桌面”屏幕的尺寸。

☐ /edit "ConnectionFile"

打开指定的 .rdp 文件进行编辑。

□ /migrate

将使用“客户端连接管理器”创建的旧连接文件迁移到新的 .rdp 连接文件中。

4.3.4 在 Centos5.5 下如何正确配置网关

由于现在的服务器都安装了防火墙及 SNAT，所以网关配置的正确与否就很重要了。我们在正确配置网关后还应该用命令 `tracert` 来检测其是否生效，如下所示：

```
vim /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
BOOTPROTO=static
HWADDR=00:22:15:c9:3c:0f
IPV6INIT=yes
IPV6_AUTOCONF=yes
ONBOOT=yes
NETMASK=255.255.255.0
IPADDR=192.168.4.45
GATEWAY=192.168.4.3
TYPE=Ethernet
```

首先，我们用命令检测其是否生效，可用 `netstat -rn` 或 `route -n` 命令，如下所示：

```
netstat -rn
Kernel IP routing table
Destination    Gateway         Genmask         Flags   MSS Window  irtt Iface
192.168.4.0    0.0.0.0         255.255.255.0   U        0 0          0 eth0
169.254.0.0    0.0.0.0         255.255.0.0     U        0 0          0 eth0
0.0.0.0        192.168.4.3     0.0.0.0         UG       . 0 0        0 eth0
```

大家注意带有 UG 标识的这一行，它表示 192.168.4.3 为机器的网关地址。在 Centos5.5 中还可以用命令 `tracert` 查看外网 IP 地址来判断最近的下一跳地址是否为你的默认网关，此命令也适用于 FreeBSD8.1，如下所示：

```
tracert www.163.com
tracert to www.163.com(210.51.213.180), 30 hops max, 40 byte packets
 1  192.168.4.3 (192.168.4.3)  0.584 ms  0.638 ms  0.703 ms
 2  220.249.72.129 (220.249.72.129)  4.099 ms  7.870 ms  11.611 ms
 3  218.104.109.77 (218.104.109.77)  6.787 ms  6.792 ms  6.933 ms
 4  218.104.110.193 (218.104.110.193)  3.158 ms  3.159 ms  3.148 ms
 5  218.106.127.82 (218.106.127.82)  3.804 ms  3.800 ms  3.868 ms
 6  218.104.110.82 (218.104.110.82)  3.841 ms  3.202 ms  3.551 ms
 7  220.249.83.130 (220.249.83.130)  3.614 ms  3.455 ms  3.514 ms
 8  210.51.213.180 (210.51.213.180)  2.721 ms  3.007 ms  3.487 ms
```

其中，第一行即为系统现在默认的网关地址。

4.3.5 VMware 的机器应该如何配置自动对时

我将成功的项目案例放在 VMware Server 上，以便在线上环境对客户进行项目演示。可是我发现虚拟

的 Centos5.5 x86_64 的时间比真实的时间总是慢一点，查阅了网上的相关资料，发现具体原因如下：

Linux 2.6 的核心里把系统计时器的频率加高到 1000Hz 了，VMware 没办法每隔 1ms 就报一次信号给 guestOS，所以 guestOS 里的 Linux 2.6 无法接到计时器的信号。本来这也不应该造成问题，不过 Linux 2.6 的核心在处理“tick loss”这个程序时有问题，以至于 guestOS 里的 Linux 2.6 系统时间走一秒然后慢一秒，也就是说外面走了两秒里面只走了一秒。网上提供的具体解决方案是修改内核及安装 vmware-tools 软件包。

我觉得很麻烦，所以采用脚本自动对时的方法解决了此问题，步骤如下：

1) 直接用脚本解决不能对时的问题，脚本/root/ntpdate.sh 的内容如下：

```
#!/bin/bash
while :
do
/usr/sbin/ntpdate ntp.api.bz >/dev/null 2>&1
sleep 5
done
```

放入后台执行 nohup sh ntpdate.sh，不要忘了 nohup。

2) 当然还是得 Crontab 出马，在这里要注意一下脚本与 Crontab 的区别，脚本可以控制到秒，而 Crontab 只能控制到分。/etc/crontab 的内容如下：

```
**/5*** root/usr/sbin/ntpdate ntp.api.bz >/dev/null 2>&1
```

效果很明显，每隔 5 分钟系统自动对时一次，这样系统时间的精确度就大大提高了。

4.3.6 防火墙初始化的注意事项

我在配置电信机房中某一台服务器的 iptables 时，不小心设置了某一项错误参数（现在回忆应该是 iptables-P OUTPUT DROP），结果锁定了 ssh 会话，导致连不上服务器。现将解决办法推荐给大家，这对于生产环境的服务器来说还是极其有用的，建议大家参考。

我们配置一个计划任务 Crontab，每 5 分钟运行一次，如下所示：

```
* /5*** root/bin/sh/root/firestop.sh
```

firestop.sh 的内容如下：

```
service iptables stop
```

这样即使你的脚本存在错误的设置规则，也不至于将你锁在计算机外而无法与服务器连接，让你可以放心大胆地调试你的脚本，这是生产环境下逼出来的办法。

4.4 系统维护时的注意事项

无论是自己的内网开发机器还是线上的生产机器，我们在操作时都应该谨慎，不然系统很容易发生崩溃的情况。就算我们能用备份很快恢复系统，在恢复时间之内，我们也会损失大量的数据，而且如果万一恢复不了，那就是灾难性的崩溃了。

4.4.1 尽量源码安装，谨慎操作 yum

Centos5.5 下的 yum 安装软件使用起来非常方便，它可以自动解决 rpm 包之间的依赖关系，比

如我们安装 rpm 包 A，但包 A 需要包 B 的库，包 B 又需要包 C，包 C 又需要包 D，如果用 rpm -vih A，我们就还需要安装包 B、C、D，但如果我们用 yum-y install A 的话，系统会自动将 A 需要的软件包 B、C、D 全安装上，这非常方便。但 yum 也是双刃剑，在提供方便的同时也是很危险的。我们有台机器一不小心用 yum remove 删除了一个软件包，由于此软件包的依赖性太多，结果同时删除了 300 多个软件包，直接导致系统崩溃了。我现在的操作是，形成标准化的流程，我在进行所有的 yum remove 操作时会在最小化安装 Centos5.5 后立即执行。比如我要用 LNMP 环境的话，先删除系统自带的 php (yum remove php)。正式上线后，基础库的安装我会用 yum install，软件包的安装我会选择用源码安装。除此之外，建议大家不要用 yum remove，尤其生产环境下的机器更不要用。

4.4.2 服务器硬件改动进入了 Emergency 模式

一位同事在处理一台 RHEL5.1 服务器时，从机器上移走了一块硬盘，然后就直接启动机器，忽然发现系统进入了 Emergency 模式。我问他：你改动硬件了没有？他说他移走硬盘后就直接启动了，这不是跟 Windows Server 2003 一样吗，有什么问题？我先讲解了一下 Linux 系统下/etc/fstab 的作用及语法，最后边操作边告诉他整个解决步骤，如下：

- 1) 在 Emergency 模式下输入 root 密码进入单用户模式。
- 2) 修改 fstab 文件时会出现 Read-only file system，重新将其装载成读写模式，命令如下：

```
mount -o remount,rw /
```

这句话的作用是将/分区设置成可读写。

- 3) 编辑/etc/fstab 文件，将移除的硬盘用 # 号屏蔽掉，然后重启服务器，故障解除。

4.4.3 如何以普通用户的身份编辑无权限的文件

如果普通用户用 Vim 编辑 nginx.conf 等配置文件，保存的时候会提示：没有 Root Permission。解决办法是在保存时加上以下命令：

```
:w ! sudo tee %
```

这条命令的含义是把当前编辑的文件内容当做标准输入，并输入到命令 sudo tee 文件名里去。也就是将 sudo 保存为当前文件名，这是一个相当管用的命令，尤其适合 FreeBSD 和 Debian 系统（我经常忘记自己原来不是 root 了）。不过这个命令不太好记，我采用谐音记忆的方法记住了它（即“我要速度去找缘分”）。当然了，大家也可以用自己的方法来记忆。

4.4.4 在 Linux 下配置最大文件打开数的方法

在 Linux 下部署应用（特别是 Squid 服务器）的时候，有时会遇上 Socket/File: Can't open so many files 问题，这也会影响服务器的最大并发数。其实 Linux 是有文件句柄限制的，而且 Linux 默认不是很高，一般都是 1024，如果是生产服务器用，很容易就会达到这个数量。

1. 查看方法

我们可以用 ulimit -a 来查看所有限制值，命令如下所示：

```
ulimit -a
core file size      (blocks, -c)0
```

```

data seg size      (kbytes, -d)unlimited
max nice           (-e)0
file size          (blocks, -f)unlimited
pending signals    (-i)4096
max locked memory  (kbytes, -l)32
max memory size    (kbytes, -m)unlimited
open files         (-n)1024
pipe size          (512 bytes, -p)8
POSIX message queues (bytes, -q)819200
max rt priority    (-r)0
stack size         (kbytes, -s)10240
cpu time           (seconds, -t)unlimited
max user processes (-u)4096
virtual memory     (kbytes, -v)unlimited
file locks         (-x)unlimited||<

```

其中 open files (-n) 1024 是 Linux 操作系统对一个进程打开的文件句柄数量的限制（也包含打开的套接字数量，可影响 MySQL 的并发连接数目）。这个值可用 ulimit 命令来修改，但 ulimit 命令修改的数值只对当前登录用户目前使用的环境有效，系统重启或用户退出后就会失效。

系统的总限制是在这里：/proc/sys/fs/file-max。我们可以通过 cat 查看其目前的值，修改/etc/sysctl.conf 也可以控制这个数值。

另外还有一个知识点，使用/proc/sys/fs/file-nr 可以看到整个系统目前使用的文件句柄数量。

查找文件句柄问题的时候，还有一个很实用的程序 lsof。可以很方便看到某个进程打开了哪些句柄，也可以看到某个文件/目录被什么进程占用了。

2. 修改方法

若要令修改 ulimits 的数值永久生效，则必须修改配置文档，可以将 ulimit 修改命令放入/etc/profile 里面。不过，这个方法实在不是很方便。还有一个方法是修改/etc/sysctl.conf。我修改后测试过，发现它不会改变用户的 ulimits -a，只是/proc/sys/fs/file-max 的值变了。

而我以前的做法是修改/etc/security/limits.conf，里面有很详细的注释，文件内容修改如下：

```

* soft nofile 32768
* hard nofile 65536

```

这样可以将文件句柄限制统一改成软 32768，硬 65536。配置文件最前面的内容是指 domain，设置为星号代表全局。另外也可以针对不同的用户做出不同的限制。

注意 其中的硬限制是实际的限制，而软限制是警告限制，它只会给出警告。其实 ulimit 命令本身就分软硬限制，加-H 就是硬限制，加-S 就是软限制。默认显示的是软限制，如果运行 ulimit 命令修改时没有加上-H 或-S，就是两个参数一起改变。

不过，这种做法仅对当前配置有效，如果重启系统的话，则又完全失效了。正确的做法应该是：编辑/etc/rc.local 文件，在其后添加如下内容：

```
ulimit -SHn 65535
```

如果有那种即时监控的脚本，将此行写进去也是可以的。这样可以保证每次重启后，limit 的值

也是 65535，我们可以通过如下脚本 `nginx_limit.sh`（不要用 `ulimit -a`）来查看 Nginx 进程能打开文件的最大数，脚本内容如下：

```
for pid in `ps aux |grep nginx |grep -v grep |awk '{print $2}'`
do
cat /proc/MYM{pid}/limits |grep 'Max open files'
done
```

如果是非 Nginx 的其他进程，可以用下面的方法来判断，如下所示：

```
cat /proc/PID/limists |grep "Max open files"
```

PID 为进程的 pid 号，我们可以通过 `ps` 命令来得到，这种方法可以精确得到当前进程打开文件的最大数。

4.4.5 在 Crontab 下运行 PHP 程序的正确方法

在 Centos5.5 下，有人是这样配置 Crontab 定时执行 PHP 程序的：先写一个 Crontab，定时执行某段 shell 脚本，然后脚本再执行 PHP 程序。这样做明显效率不高。最简单的方法是，直接在 `/etc/crontab` 里执行 PHP 程序，命令如下：

```
* /5 * * * root php /home/andrewy/test.php
```

在最小化安装 Centos5.5 时已安装了 Perl 和 Python，如下所示：

```
rpm -q perl
perl -5.8.8-27.el5
rpm -q python
python -2.4.3-27.el5
```

所以，Perl 和 Python 程序在 Centos5.5 下就不需要再安装 Perl 或 Python，如果要将其放在 Crontab 里按时执行的话，可以按照以上实例类推。

4.4.6 在 Crontab 下正确防止脚本运行冲突

如果某脚本要运行 30 分钟，可以在 Crontab 里把脚本间隔设为至少一小时来避免冲突。而比较糟的情况是可能该脚本在执行周期内没有完成，接着第二个脚本又开始运行了。如何确保只有一个脚本实例运行呢？一个好用的方法是利用 `lockf`（FreeBSD8.1 下为 `lockf`，Centos5.5 下为 `flock`），在脚本执行前先检测能否获取某个文件锁，以防止脚本运行冲突。

`lockf` 的参数如下。

- `-k`：一直等待获取文件锁。
- `-s`：silent，不发出任何信息，即使拿不到文件锁。
- `-t seconds`：设定 timeout 的时间是 seconds 秒，如果超过时间，则自动放弃。

以下 Crontab 计划任务执行前，需获取临时文件 `create.lock` 的文件锁，此项 Crontab 计划任务的内容如下：

```
* /10 * * * * (lockf -s -t 0 /tmp/create.lock /usr/bin/python /home/project/cron/create_tab.py
>> /home/project/logs/create.log 2>&1)
```

若第一个实例在 10 分钟内没有运行完，第 2 个实例不会运行。我以前是通过 shell 脚本来解决这个问

题的, 比如用 `while...do` 循环, 然后放在后台执行。但后来发现其实用 `flock` 或 `lockf` 方法更为简单。

4.5 紧急处理线上服务器故障的办法

很多时候, 网站或业务系统的服务器出现了故障, 我们必须紧急修复, 保证网站或业务系统能够使用。一般我们会遇到哪些系统故障, 又该如何来处理呢? 大家可以看看以下内容。

4.5.1 更改 Administrator 密码导致计划任务无法执行

1. 问题描述

公司有位系统管理员离职了, 他曾负责管理不少 Windows Server 2003 服务器, 于是负责安全的部门要求接手的系统管理员更改 Administrator 密码。粗心的系统管理员急急忙忙地更改了 Windows Server 2003 的 Administrator 密码, 却发现 Windows Server 2003 的计划任务 (即 Scheduled Tasks) 全都执行不了。出现这个问题其实是因为 Windows Server 2003 的计划任务都要求输入正确的 Administrator 密码。

2. 解决办法

大家养成好习惯, 每次更改完 Windows Server 2003 密码后一定要检查一下计划任务是否能正常执行, 否则很容易导致公司的重要业务执行不了, 进而影响整个网站的运维及业务。希望此问题能引起大家的注意。

4.5.2 FreeBSD8.1 下的 sudoer 文件意外损坏

1. 问题描述

同事远程处理机房的某台 FreeBSD8.1 机器, 本来想加一个具有 sudo 权限的用户进去, 所以编辑了 `/etc/sudoer` 文件, 却不小心多加了一个 `.`, 然后直接保存退出了。结果悲剧发生了: 由于 sudoer 文件损坏, 所有具有 sudo 权限的用户均不能切换到 root 模式下工作。而 FreeBSD8.1 与 Centos5.5 等 Linux 系统不同的是, 它默认是不允许 root 远程连接的。

2. 解决方法

出现这个问题只有请人到机房去处理了。建议在更改这些危险文件前, 增加在 FreeBSD8.1 下 root 能远程 SSH 的控制选项。另外, 在机器托管到 IDC 机房前就做好这些账户添加的工作, 机器上线以后就不要再添加用户了。

4.5.3 Centos5.5 的 root 密码被恶意篡改

在 Centos5.5 系统下, 用户基本都是由 `su-` 切换到 root, 不过这时有一个问题, 假如有多人知道 root 的账号和密码, 那么如果出现问题, 你就不知道到底是谁干的坏事, `log` 里记录的都是 root 干的。我的开发服务器的 root 密码就被人神不知鬼不觉地改了, 因为是台新机器, 暂时还没有分配 sudo 权限的用户, 所以只有到机房去改密码了。同事时常有清空 memcached 缓存的需求, 我不敢把 root 密码给他们。其实, 这时候可以给他们的账户 sudo 权限, 这样就可以通过 `log` 或 `last` 等命令查询他们做的事情。

实际部署方法如下: 我们可以建立一个 admin 组, 让这个组的用户权限和 root 一样。我们公司有 4 个开发小组, 所以准备给每一个小组一个账户, 密码由他们各自保管, 以后考虑将权限细化分

配。具体步骤如下。

1) 修改 sudoers 文件, 即/etc/sudoers 文件, 在最后加入以下内容:

```
%admin ALL = (ALL) ALL
```

2) 添加 admin 组, 命令如下:

```
groupadd admin
```

3) 添加用户, 这里分为 4 个组, 即开发一组至开发四组, 添加如下命令:

```
useradd yuhongchun (开发一组)
useradd wangxiaona (开发二组)
useradd zhanghua (开发三组)
useradd tangyihe (开发四组)
```

密码由 4 个小组自己设置和保管, 相互之间不知道其他组的密码。

4) 添加用户到 admin 组, 命令如下:

```
usermod -a -G admin yuhongchun
usermod -a -G admin wangxiaona
usermod -a -G admin zhanghua
usermod -a -G admin tangyihe
```

这样每个开发人员就都有 sudo 权限了, 而每个用户所做的事情基本可以通过日志查询到。

4.5.4 bash 损坏该如何正确处理

故障描述:

在 FreeBSD8.0 的 jail 虚拟机 192.168.21.36 上安装一个软件时不小心将其依赖的库文件 libintl.so.8 丢失了, 导致所有以 bash 为 shell 的用户不能登录。由于大家都喜欢在此机上用 bash, 所以均将其配置成了默认的 shell, 也就是说所有的用户都不能登录了。系统报错如下:

```
/libexec/ld-elf.so.1: Shared object "libintl.so.8" not found, required by "bash"
Connection to 192.168.21.36 closed.
```

解决方法如下所示:

1) 用单用户模式进入系统。

2) 扫描磁盘 (此步操作是安全的), 命令如下:

```
fsck -y
```

3) 将文件系统重新挂载, 命令如下:

```
mount -a
```

4) 将 root 的默认 shell 切换到 sh, 命令如下:

```
chsh -s sh
```

5) 其实进行到了第四步, 我们就可以用 root 以 sh 模式进入系统了, 但为了完美地解决问题, 这时候可以安装一下 bash, 命令如下:

```
pkg_add -r -v bash
```

重启后一切正常，故障排除。

4.5.5 正确操作 nohup 让程序始终在后台运行

我的 Nginx 负载均衡器监控 Nginx 进程的脚本 `nginx_pid` 需要放入后台不间断地运行，所以想用命令 `/bin/sh/data/nginx_pid.sh &` 来达到此目的，在输入完命令后我就关闭了终端，可再次登录终端时发现此程序并没有运行。忽然想起可能是因为没有使用 `nohup` 命令，试了试，果然如此，带上 `nohup` 命令后就正常了。

我们的很多程序只是普通的程序，即使它们使用 `&` 结尾，如果终端关闭，那么程序也会被关闭。为了能够在后台运行，我们需要使用 `nohup` 这个命令，原程序的标准输出被自动改到当前目录下的 `nohup.out` 文件里，起到了 `log` 的作用。

但是有时候这样做会有问题：如果把终端关闭，进程也会被自动关闭。查看 `nohup.out` 可以看到在关闭终端的瞬间服务自动关闭了。

产生此问题的原因是：虽然 `shell` 中提示了 `nohup` 成功，但还是需要按键盘上的任意键退回到 `shell` 输入命令窗口，然后通过在 `shell` 中输入 `exit` 来退出终端，而不是每次在 `nohup` 执行成功后直接关闭终端。

这个错误许多朋友（包括我）容易忽视，希望大家在工作中注意。

4.5.6 负载均衡器出现故障

在主 Nginx 负载均衡器上，由于 `nginx.conf` 配置文件有误，导致客户不能正常访问网站。网站架构用的是 Nginx + Keepalived，这时我们可以紧急停掉主 Nginx 上的 Keepalived，让从机接管，等网站稳定后，再来修复主 Nginx 负载均衡器。

注意 其实单纯停掉 Nginx 是解决不了问题的，因为我们的 VIP 此时还挂在主 Nginx 上面。所以要 让从 Nginx 生效，停掉主 Nginx 上面的 Keepalived 是最好的方法。

另外如果 LVS/DR 模式下的 LVS 负载均衡器都停止工作了，其实也不难进行应急处理。通过修改 DNS 的 A 记录，我们可以把先前主机名对应的 VIP 改成真实服务器的 IP 地址，使服务迅速恢复起来，从而赢得时间处理负载均衡器的故障。

总而言之，线上环境的服务器排障要求我们在最短的时间内解决问题，这其实也是对系统管理员经验和能力的考验。我们平时应该总结在线上环境中容易出现的故障，做到未雨绸缪；平时在维护网站或系统时应该着重注意容易崩溃的部分，多花些精力和时间研究它们；如果觉得压力过大，单机负载承受不了，可以考虑采用集群的方法来处理。

4.6 检查机房应注意的位置和细节问题

如果我们的服务器都是 IDC 托管服务器的话（即服务器不在自己的机房内），建议采用如下做法：

- 服务器中最容易坏的是风扇，如果是电信机房要注意多检查一下，尤其是生产环境下的重要服务器，建议都配置成集群环境，这样也方便进行检查或升级；如果是自己的内网服务器机房，平时注意将机房温度控制在 17 摄氏度。
- Dell 系列或 HP 系列的机器中 RAID 卡放电是正常现象，如果因此有 Nagios 报警短信和邮件

也是正常的。

- 有时间多巡视机房，检查服务器的硬盘灯指示情况，数据量频繁读写时极易发生硬盘故障。
- 注意网线不要松脱了，不然你使用的 Heartbeat + DRBD 服务器就很麻烦了。
- 平时如果有时间和机会，可以做一些关于 Keepalived 和 Heartbeat 的模拟故障实验，保证我们的网站或系统的高可用性。
- 平时可以多学习一些网络相关的知识，并研究一些疑难问题，特别是硬件防火墙的端口映射问题，有时很多问题是因网络问题引起的。

4.7 系统维护时应注意的非技术因素

我们在平时进行系统维护时，除了技术方面的因素，我们还要注意一些非技术的因素，因为它们也是安全隐患，如果大意的话，极有可能影响网站运行。我特地归纳了几点，如下所示：

- 在机房巡视时，我发现很多管理员（尤其是刚刚参加工作的），特别喜欢用 root 进行维护，并且不退出就离开了，这种做法是极其危险的。因为如果机房的机器多，会涉及几个部门的 System Admin，极容易出现误操作的情况。所以正确的做法应该是操作完成后应立即用 Ctrl + A 快捷键退出，然后锁上机房门。另外，我建议重要的服务器不设监视器和键盘，一切操作都由远程处理，除非有特别需求，才用这些东西进行维护。
- 无论是网站还是系统，要正常运行都会涉及很复杂的工作，比如需要防火墙、程序、应用、数据库等都能够正常运行；在正式上线后还要考虑复杂的大量并发和安全问题，在测试环境下能够顺利运行的程序或脚本，未必能在线上正常运行。
- 如果是刚刚接触系统管理工作的新人，我建议先从 sudo 用户用起，在熟练到一个级别后再用 root 进行 Linux/Unix 的管理工作。sudo 虽然麻烦些，但在一些危险性或毁灭性的操作前还是有一定的预警或防范作用的。
- 我和同事在 Linux/Unix 服务器上操作配置文件时养成了一个好习惯，即从不删除任何文件。当需要更改重要文件时，我们一般会先备份 .bak 文件，然后再编辑。如果是配置文件的某个语段，我们一般会用#注释掉，而不用清除。大家以后慢慢会发现，这个好习惯会给你的工作带来许多便利，消除危险因素。
- 如果是用 Xshell 3.0 来操作线上环境的服务器，而且是多窗口多服务器的操作，我建议在做完操作后就立即退出当前窗口，这也是消除不安全因素的一种做法。因为当测试环境和线上环境的服务器都在 Xshell 3.0 窗口时，极容易出现误操作的情况。

4.8 小结

在处理生产环境下的服务器故障时，应该本着安全的原则，在尽可能短的时间内处理问题，恢复服务器的正常状态。平时的系统维护工作，在多备份的前提下，应该也本着谨慎小心的原则，减少失误，毕竟我们不可能将每一项重要应用服务都做成集群模式，有些重要业务应该准备备份服务器，这样在出现问题时可以手动将备份的服务器换上去，然后再来解决故障服务器上的问题。另外，我们在用 Xmanager 3.0 远程管理多台线上服务器时，应该养成的好习惯是操作完一台就退出这一台的窗口，以免将其当成了测试服务器，从而发生误操作的悲剧。希望大家能在日常的系统维护工作中注意这些。



第 5 章 生产环境下的SHELL脚本

- 5.1 Vim 的基础用法及进阶心得
- 5.2 Sed 的基础用法及实用举例
- 5.3 基础正则表达式
- 5.4 Linux 下强大的查找命令 find
- 5.5 汇总 Linux/Unix 下的 bash 快捷键
- 5.6 生产环境下的 SHELL 脚本分类
- 5.7 小结

接触 Linux/Unix 系统六七年了，SHELL 脚本已完全融入我的生活中。虽然 SHELL 脚本只是一个简单的解释型语言，不会受到开发人员的重视，但对于我们系统管理员来说它有着举足轻重的作用，它可以帮助我们简化日常的工作并减少工作量，成为我们的瑞士军刀。我们在系统维护工作中用 SHELL 脚本常常能比用 C 或 C++ 语言编写的程序更快地解决相同的问题。此外，SHELL 脚本具有很好的可移植性，有时跨越 Unix 与 POSIX 兼容的系统，仅需略作修改，甚至不必修改即可使用 SHELL 脚本。

在日常工作中 SHELL 脚本能帮助我们做什么呢？

1) 配合 Crontab 帮助我们定时执行任务，就像 MS 的计划任务一样。很多朋友向我反映说 Crontab 不能做秒级的计划任务，其实只要写一个 SHELL 脚本，用 while..do..done 循环后放入后台执行就可以实现秒级的计划任务。不过，为了避免造成死循环，记得要加入 sleep 5 的代码，这样程序会在执行完毕后休息 5 秒，也可以说每 5 秒钟就执行了一次程序。

2) 配合 PHP 等开发程序进行日常的开发维护工作，比如我们的 SVN 发布程序就是通过 PHP + SHELL 来实现的。

3) 配合 iptables 可形成方便和安全的 iptables 脚本，可保护我们的主机安全。

4) 它可以成为工程人员的工具箱，用来解决日常 Linux/Unix 环境中遇到的相关问题，例如文本过滤筛选、系统日志分析等。

5) 可以写强大的系统性能和状态监控脚本，并配合 Keepalived 来实现系统的高可用。

6) 备份和 rsync 同步重要服务器资料，这是 SHELL 的基本功能。

7) 自动化安装系统环境，规范化操作，缩减项目实施的时间和误差。

SHELL 的强大和其他未挖掘功能需要我们在日常工作发现和总结。下面我以我们的线上环境为平台，跟大家分享一下线上服务器的 SHELL 脚本。

另外，在跟大家分享 SHELL 脚本之前，我先总结一下在 Centos/FreeBSD 下的编辑工具 Vim 及流编辑器 SED 的使用，还有 Bash 命令行快捷键方式及正则表达式，我们可以用它们来编辑 SHELL 脚本。结合这些我们可以写出强大的 SHELL 脚本，如果熟练地掌握和运用它们，我们的工作效率会得到很大的提高。本章内容适合有 SHELL 基础的计算机专业学生、PHP 或 Java 开发人员及 Linux/Unix 系统工程师学习。另外，在这里我要感谢所有给我提供生产环境下脚本的朋友们，请允许我在这里介绍崔晓辉先生，他是大众网高级系统管理员，擅长网站系统架构和 Unix 技术。另外这里也要感谢我的同事 Ritto，感谢他们提供了大量实用的生产环境下的 SHELL 脚本，让我们的工作变得如此轻松而有效率。

5.1 Vim 的基础用法及进阶心得

vi 作为开源系统的默认编辑器，现在为越来越多的人了解和熟悉，而 Vim 作为 vi 的升级版本，在功能上又有明显的提高。由于它方便实用，现在越来越多的开发人员也喜欢用其作为代码编辑工具了。vi 更符合 Unix 传统，它通过管道机制和系统内的各种积木工具打交道，它讲究的是和系统内的工具程序协作来完成用户的任务。Vim 是 vi 最受欢迎的变种之一，它除了继承了 vi 迅捷的编辑方式外，在功能方面也已经比原始的 vi 强大很多。它现在是 Centos5.5 及 FreeBSD8.1 下首选的强大编辑器之一。下面我归纳总结了 Vim 的基础功能，如表 5-1 所示。

表 5-1 Vim 基础用法列表

命令模式	光标移动
h 或向左方向键	光标向左移动一个字符
j 或向下方向键	光标向下移动一个字符
k 或向上方向键	光标向上移动一个字符
l 或向右方向键	光标向右移动一个字符
Ctrl + f	屏幕向前翻一页 (常用)
Ctrl + b	屏幕向后翻一页 (常用)
Ctrl + d	屏幕向后翻半页
Ctrl + u	屏幕向前翻半页
+	光标移动到非空格符的下一列
-	光标移动到非空格符的上一列
n < space >	按下数字键后再按空格键, 光标会向右移动这一行的 n 个字符。例如 20 < space > 即光标向右移动 20 个字符
0 (HOME)	(是数字 0) 移动到这一行的第一个字符处 (常用)
\$(END)	移动到这一行的最后一个字符处 (常用)
H	光标移动到当前屏幕最上方的那一行
M	光标移动到当前屏幕中央的那一行
L	光标移动到当前屏幕最下方的那一行
G	光标移动到文件的最后一行
nG	移动到当前文件的第 n 行。例如 20G, 表示移动到当前文件的第 20 行 (可配合: set nu)
n < Enter >	光标向下移动 n 行 (常用)
命令模式	查找与替换
/word	在光标之后查找一个名为 word 的字符串 (常用)
? word	在光标之前查找一个名为 word 的字符串
: n1, n2s/word1/word2/g	在第 n1 与 n2 行之间查找 word1 这个字符串, 并将该字符串替换为 word2 (常用)
: 1, \$s/word1/word2/g	在第一行与最后一行之间查找 word1 这个字符串, 并将该字符串替换为 word2 (常用)
: 1, \$s/word1/word2/gc	在第一行与最后一行之间查找 word1 这个字符串, 并将该字符串替换为 word2, 且在替换前显示提示符让用户确认 (confirm) (常用)
一般模式	删除、复制与粘贴
x, X	x 为向后删除一个字符, X 为向前删除一个字符 (常用)
Nx	向后删除 n 个字符
Dd	删除光标所在的那一整行 (常用)
ndd	删除光标所在列的向下 n 列, 例如, 20dd 表示删除 20 列 (常用)
d1G	删除光标所在行到第一行的所有数据
dG	删除光标所在列到最后一行的所有数据
Yy	复制光标所在行 (常用)
nyy	复制光标所在列的向下 n 列, 例如, 20yy 表示复制 20 列 (常用)
y1G	复制光标所在列到第一列的所有数据
yG	复制光标所在列到最后一列的所有数据
p, P	p 为复制的数据粘贴在光标下一列, P 为粘贴在光标上一列 (常用)
J	将光标所在列与下一列的数据结合成一列
U	恢复前一个动作 (undo)

(续)

编辑模式	
i, I	插入：在当前光标所在处插入输入的文字，已存在
a, A	添加：由当前光标所在处的下一个字符开始输入，已存在的字符会向后退（常用）
o, O	插入新的一行：从光标所在行的下一行行首开始输入字符（常用）
r, R	替换：r 会替换光标所指的那一个字符；R 会一直替换光标所指的字符，直到按下 Esc 键为止（常用）
Esc	退出编辑模式，回到一般模式（常用）
命令行模式	
: w	将编辑的数据写入硬盘文件中（常用）
: w!	若文件属性为只读，强制写入该文件
: q	退出 vi（常用），快捷方式为 Shift + ZZ
: q!	若曾修改过文件，又不想保存，使用！为强制退出不保存文件，快捷方式为 Shift + ZQ
: wq	保存后退出，若为：wq!，则表示强制保存后退出（常用）
: w [filename]	将编辑数据保存为另一个文件（类似另存新文档）
: r [filename]	在编辑的数据中，读入另一个文件的数据。即将 filename 这个文件内容加到光标所在行的后面
: set nu	显示行号，设定之后，会在每一行的前面显示该行的行号
: set nonu	与 set nu 相反，为取消行号显示
: set nohlsearch	可取消高亮，可编辑/etc/vimrc 来取消所有高亮
n1, n2 w [filename]	将 n1 到 n2 的内容保存为名为 filename 的文件

另外，FreeBSD8.1 中默认的 Vim 并不是太好使用，推荐大家采用 Vim 模板的形式编辑当前用户的 .vimrc 文件，让 Vim 可以更方便地为我们工作，模板文件如下：

```
set nobackup
set noswapfile
set nohlsearch
set nonumber
set cindent
set autoindent
set shiftwidth=2
set tabstop=2
set softtabstop=2
set expandtab
set ruler
set mouse=v

syntax on
```

1. 常用设定

下面是一些 Vim 中常用的设定及其具体含义。

- set nobackup：不要备份文件，使用 backup 备份文件（原文件加后缀 ~）。
- set noswapfile：不生成 .swap 文件。我在编辑文件之前总有个习惯，一般会将其原文件进行 .bak 备份，然后才很放心地在相关文件上进行修改。如果操作谨慎的朋友建议不要此项。

- `set number`: 显示行号。
- `set cindent`: 设定 c 风格缩进, 可使用 `nocindent` 取消设置。
- `set autoindent`: 设定自动缩进, 每行缩进与上一行相等, 可使用 `noautoindent` 取消设置。
- `set shiftwidth = 2`: 设定缩进为两个空格。
- `set tabstop = 2`: 设定制表符为两个空格。
- `set softtabstop = 2`: 设定软制表符为两个空格。
- `set expandtab`: 缩进和 (软) 制表符使用空格替代, 可用 `noexpandtab` 取消设置。
- `set ruler`: 显示光标所在行列号。
- `set mouse = v`: 如果我们用 `set mouse = a` 启动了所有模式, 屏蔽了鼠标的右键功能, 那么就可以用此语法让其在 Vim 可视化模式下也能使用鼠标右键复制功能。
- `syntax on`: 启动语法高亮。

下面是其中涉及的名词术语的相关解释:

□ `cindent`

使用 C 语言的缩进方式, 根据特殊字符如 “{”、“}”、“:” 和语句是否结束等信息自动调整缩进。在编辑 C/C++ 等类型文件时自动设定。

□ `softtabstop`

软制表符宽度, 设置为非零数值后使用 Tab 键和 Backspace 键时光标移动的格数等于该数值, 但实际插入的字符仍受 `tabstop` 和 `expandtab` 控制。

我们在按照如上模板文件配置了 Vim 后应该会发现 Vim 比以前好用多了。我目前将其模板文件应用于线上 FreeBSD、Centos 及 Debian 系统中。当然了, 这些也只是 Vim 的基础配置, 朋友们可以根据自身的习惯配置更个性化的 Vim。

2. 日常系统维护工作中的 Vim 使用心得

1) FreeBSD8.1 下的 Vim 与 Centos5.5 下的 Vim 是不一样的, 建议优化各用户下的 `.vimrc` 文件。我现在的习惯做法就是随身带 `.vimrc` 配置模板, 哪台机器上有我的账户, 我就直接导入到哪台机器的 `/home/andrewy` 的账户下, 这样用起来就得心应手了。

2) 以上用法仅仅满足 System Administrator 的基础工作, 如果是开发人员, 强烈建议继续深入研究 Vim 的高级语法及插件。特别是作为 PHP 和 Java 的开发人员, 用 Vim 的基本功能配合插件编辑程序代码会是件很轻松的事情。

3) 如果习惯用图形界面使用 MS 的朋友, 可以下载一个 `gvim` 来学习及工作。

4) 在加深 Vim 学习的同时, 也建议加强对 Sed 的理解和学习, 尽量用 Vim + Sed 的方式来完成日常中的编辑工作, 例如 SHELL、Python 和 Perl 脚本, 以及和系统相关的配置文件。

5) 尽量用 h、j、k、l 来进行左下上右操作, 这远远比你用键盘的方向键更有效率, 但这是大家很容易忽视的一个问题, 有的老 Unix 机器不支持方向键。

6) Vim 不仅仅是一个编辑器, 你完全可以用它来查看服务器的配置文件 `.conf`、`.php`、`.jsp` 文件及 `.sh` 等程序文件。如果用得很熟练, 也可以查看日志文件, 注意最后不保存退出即可。

7) 这里只介绍两个有用的插件 `taglist` 和 `ctags` (事实上我也基本只用这两个)。用过 eclipse 的人可能会对按住 Ctrl 键点击程序中的函数、变量等可自动跳转到其定义处的功能赞叹不已, 其实, 这个功能 Vim 也可以实现, 用的就是 `ctags`。Vim 的插件功能非常强大, 希望有兴趣的朋友

深入了解一下。

8) 我现在利用 gvim 辅助 Excel 等系统处理数据时, 主要利用了正则替换、列模式等功能。比如把“2006-6-30”替换为“2006.6.30”或“2006.06.30”。

9) 我现在用得较多的 gvim 主要是用来编辑 SHELL 文件和处理文本文档的(比如我自己的电子文档), 感觉效果非常理想。

10) Emacs 我也接触过, 感觉不是太好上手。如果是刚接触 Linux/Unix 系统的话, 推荐用 Vim。

11) C 文件也是可以用 Vim 来编辑的, 当然了, 用 Vim 和 gvim 来编辑 Python、Java 和 PHP 代码也是完全可行的。

5.2 Sed 的基础用法及实用举例

Sed 是 Linux/Unix 平台下的轻量级流编辑器, 日常一般用于处理文本文件。Sed 有许多很好的特性。首先, 它相当小巧; 其次, Sed 可以配合强大的 SHELL 来完成许多复杂的功能。在我看来, Sed 完全可以被看成是一个脚本解释器, 它用类似于编程的手段完成许多事情。我们完全可以用 Vim + Sed 的方式来处理日常工作中的大多数文档。

5.2.1 Sed 的基础语法格式

Sed 的格式如下所示:

```
sed [-nefr] [n1,n2] action
```

其中:

- -n: 是安静模式, 只有经过 Sed 处理过的行才显示出来, 其他不显示。
- -e: 表示直接在命令行模式上进行 Sed 的操作。是默认选项, 不用写。
- -f: 将 Sed 的操作写在一个文件里, 用的时候 -f filename 就可以按照内容进行 Sed 操作了。
- -r: 表示使 Sed 支持扩展正则表达式。
- n1, n2: 不一定需要, 选择要进行处理的行。如: 10, 20 表示在 10 ~ 20 行之间处理。

Sed 的 action (动作) 支持如下参数。

- a: 表示添加, 后接字符串, 添加到当前行的下一行。
- c: 表示替换, 后接字符串, 用它替换 n1 到 n2 之间的行。
- d: 表示删除符合模式的行, 它的语法为 sed '/regexp/d', 斜杠之间是正则表达式, 模式在 d 前面, d 后面一般不接任何内容。
- i: 表示插入, 后接字符串, 添加到当前行的上一行。
- p: 表示打印, 打印某个选择的数据, 通常与 -n 安静模式一起使用。
- s: 表示搜索, 还可以替换, 类似于 Vim 里的搜索替换功能。例如: 1, 20s/old/new/g 表示替换 1 ~ 20 行的 old 为 new, g 在这里表示处理这一行所有匹配的内容。

注意 动作最好用单引号括起来, 防止因空格导致错误。

Sed 实例如下所示 (下面所有实例在 Centos5.5_x64 下已通过):

- 1) 显示 passwd 内容, 将 2 ~ 5 行删除后显示。

```
cat -n /etc/passwd | sed '2,5d'
1   root:x:0:0:root:/root:/bin/bash
6   games:x:5:60:games:/usr/games:/bin/sh
7   man:x:6:12:man:/var/cache/man:/bin/sh
8   lp:x:7:7:lp:/var/spool/lpd:/bin/sh
...
```

2) 在第2行后面的一行加上 Hello China 字符串。

```
cat -n /etc/passwd | sed '2a Hello China! '
1   root:x:0:0:root:/root:/bin/bash
2   daemon:x:1:1:daemon:/usr/sbin:/bin/sh
Hello China!
...
```

3) 在第2行后面一行加上两行字，例如：“this is first line!” 和 “this is second line!”。

```
cat -n /etc/passwd | sed '2a This is first line! \    //使用续航符 \后按回车输入后续行
> This is second line! '                          //以'再回车结束
1   root:x:0:0:root:/root:/bin/bash
2   daemon:x:1:1:daemon:/usr/sbin:/bin/sh
This is first line!
This is second line!
3   bin:x:2:2:bin:/bin:/bin/sh
```

4) 将2~5行的内容替换成“我是大好人!”。

```
cat -n /etc/passwd | sed '2,5c 我是大好人! '
1   root:x:0:0:root:/root:/bin/bash
我是大好人!
6   games:x:5:60:games:/usr/games:/bin/sh
7   man:x:6:12:man:/var/cache/man:/bin/sh
8   lp:x:7:7:lp:/var/spool/lpd:/bin/sh
```

5) 只显示5~7行，注意 p 与 -n 的配合使用!

```
cat -n /etc/passwd | sed -n '5,7p'
5   sync:x:4:65534:sync:/bin:/bin/sync
6   games:x:5:60:games:/usr/games:/bin/sh
7   man:x:6:12:man:/var/cache/man:/bin/sh
```

6) 使用 ifconfig 列出 IP，我们只想要 eth0 的 IP 地址。可以先用 grep 取出有 IP 的那一行，然后用 Sed 去掉（替换成空）IP 前面和后面的内容，如下所示：

```
#ifconfig eth0
eth0      Link encap:Ethernet HWaddr 00:16:36:02:41:aa
          inet addr:172.30.171.35 Bcast:172.30.171.255 Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:1221198 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1125085 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1477365271 (1.3 GB) TX bytes:141539593 (134.9 MB)
          Interrupt:20
```

```
ifconfig eth0 | grep inet | sed 's/^.* addr://g' | sed 's/Bcast.*$//g'
172.30.171.35
```

'^.* addr:' 表示从开头到 addr: 的字符串, 将它替换为空, 'Bcast.*\$' 表示从 Bcast 到结尾的串, 也将它替换为空, 然后就只剩下 IP 了。

另外一种更简便的方法如下:

```
ifconfig eth0 | grep "inet addr:" | awk -F[:" "] + '{print $4}'
192.168.21.41
```

awk -F[:" "] 的意思就是以: 或空格符作为分隔符, 然后打印出第 4 列。这里有些朋友会有疑惑, 问那为什么不直接以如下方法获取 IP 呢?

```
ifconfig eth0 | grep "inet addr:" | awk -F: '{print $2}'
```

得出的结果如下:

```
192.168.21.41 Bcast
```

所以还需要再进行一步, 如下:

```
ifconfig eth0 | grep "inet addr:" | awk -F: '{print $2}' | awk '{print $1}'
```

我希望大家以这个例子好好总结一下 Sed 的经典用法。下面的方法其实是 Awk 的方法, 它也是一种优秀的编辑器, 现多用于对文本字段的列的截取。

7) 在 /etc/manpath.config 中, 将有 MAN 的设置取出, 但不要说明内容。代码如下所示:

```
cat /etc/manpath.config | grep 'MAN' | sed 's/#.*$//g' | sed '/^$/d'
MANDATORY_MANPATH      /usr/man
MANDATORY_MANPATH      /usr/share/man
MANDATORY_MANPATH      /usr/local/man
MANPATH_MAP    /bin      /usr/share/man
MANPATH_MAP    /usr/bin   /usr/share/man
MANPATH_MAP    /sbin      /usr/share/man
MANPATH_MAP    /usr/sbin  /usr/share/man
...
```

注意, # 不一定出现在行首。因此, /#.*\$/ 表示 # 和后面的数据 (直到行尾) 是一行注释, 将它们替换成空。/^\$/ 表示空行, 后接 d 表示删除空行。注意, 删除空行不能用替换方法, 因为空行替换成空后, 还是有换行符在那一行中。

以上就是 Sed 的几种常见的语法命令, 希望大家结合下面的实例, 多在自己的机器上演示, 尽快熟练掌握其用法。

5.2.2 Sed 的用法举例说明

1. Sed 的基础用法

1) 删除行首空格, 代码如下:

```
sed 's/^[ ]* //g' filename
sed 's/^ * //g' filename
sed 's/^[:space:]* //g' filename
```


2) 在行后和行前添加新行。

行后的命令如下：

```
sed 's/pattern/&\n/g' filename
```

行前的命令如下：

```
sed 's/pattern/\n&/g' filename
```

其中，& 代表 pattern。

3) 使用变量替换（使用双引号），代码如下：

```
sed -e "s/$var1/$var2/g" filename
```

4) 在第一行前插入文本，代码如下：

```
sed -i '1i\插入字符串' filename
```

5) 在最后一行插入，代码如下：

```
sed -i '$a\插入字符串' filename
```

6) 在匹配行前插入，代码如下：

```
sed -i '/pattern/ i "插入字符串"' filename
```

7) 在匹配行后插入，代码如下：

```
sed -i '/pattern/ a "插入字符串"' filename
```

8) 删除文本中空行和空格组成的行及 # 号注释的行，代码如下：

```
grep -v ^# filename | sed /^[[[:space:]]]*$/d | sed /^$/d
```

9) 要将目录/modules 下面所有文件中的 zhangsan 都修改成 list，可用如下命令（注意备份原文件），代码如下：

```
sed -i 's/zhangsan/list/g' `grep zhangsan -rl /modules`
```

2. 巧用 Vim + Sed 整理 nginxd.sh 脚本文件

我在工作中遇到了问题，于是到 Google 上下载了 Nginx 配置脚本，在复制粘贴到服务器中并运行时，发现前面的 001 ~ 100 行都有行标识符，外带空格，影响运行和美观。本来想一行行地删除，后来觉得过于麻烦，于是想到可以用 Sed 来解决问题，解决方法如下：

1) 在 Vim 里删除所有行的首数字，命令如下：

```
:%s/^[0-9][0-9]* //
```

2) 再删除所有行的首空字符，如下所示：

```
sed -i 's/^[[[:space:]]* //' nginxd.sh
```

整个 nginxd.sh 演示脚本如下，有兴趣的朋友也可以拿着练下手。

```
001    #!/bin/sh
002
```

```

003  # source function library
004  . /etc/rc.d/init.d/functions
005
006  # Source networking configuration.
007  . /etc/sysconfig/network
008
009  # Check that networking is up.
010  [ ${NETWORKING} = "no" ] && exit 0
011
012  RETVAL=0
013  prog="nginx"
014
015  nginxDir=/usr/local/nginx
016  nginxd=$nginxDir/sbin/nginx
017  nginxConf=$nginxDir/conf/nginx.conf
018  nginxPid=$nginxDir/nginx.pid
019
020  nginx_check()
021  {
022      if [[ -e $nginxPid ]]; then
023          ps aux |grep -v grep |grep -q nginx
024          if (( $? == 0 )); then
025              echo "$prog already running..."
026              exit 1
027          else
028              rm -rf $nginxPid &> /dev/null
029          fi
030      fi
031  }
032
033  start()
034  {
035      nginx_check
036      if (( $? != 0 )); then
037          true
038      else
039          echo -n "Starting $prog:"
040          daemon $nginxd -c $nginxConf
041          RETVAL=$?
042          echo
043          [ $RETVAL = 0 ] && touch /var/lock/subsys/nginx
044          return $RETVAL
045      fi
046  }
047
048  stop()
049  {
050      echo -n "Stopping $prog:"
051      killproc $nginxd
052      RETVAL=$?

```

```

053     echo
054     [ $RETVAL = 0 ] && rm -f /var/lock/subsys/nginx $nginxPid
055 }
056
057 reload()
058 {
059     echo -n $"Reloading $prog:"
060     killproc $nginxd -HUP
061     RETVAL = $?
062     echo
063 }
064
065 monitor()
066 {
067     status $prog &> /dev/null
068     if (( $? = 0 )); then
069         RETVAL=0
070     else
071         RETVAL=7
072     fi
073 }
074
075 case "$1" in
076     start)
077         start
078         ;;
079     stop)
080         stop
081         ;;
082     restart)
083         stop
084         start
085         ;;
086     reload)
087         reload
088         ;;
089     status)
090         status $prog
091         RETVAL = $?
092         ;;
093     monitor)
094         monitor
095         ;;
096     * )
097         echo $"Usage: $0 {start|stop|restart|reload|status|monitor}"
098         RETVAL=1
099 esac
100 exit $RETVAL

```

此文件还有很多变化，比如空格在开头，序列号在中间，这也可以用 Sed 来解决，只是这时又

应该写出怎样的 Sed 命令呢？这就留给大家思考吧！

3. Sed 结合正则表达式批量修改文件名

我在工作中遇到了更改文件的需求，原来某文件 test.txt 中的链接地址为：

http://www.5566.com/produce/2007080412/315613171.shtml

http://bz.5566.com/produce/20080808/311217.shtml

http://gz.5566.com/produce/20090909/311412.shtml

现要求将 http://*.5566.com 更改为/home/html/www.5566.com。我用 Sed 结合正则表达式解决之，如下所示：

```
sed -i 's/http.*\.com\/home\/html\/www.5566.com/g' test.txt
```

如果是用纯 Sed 命令，方法更简单，如下所示：

```
sed -i 's@http://[^\.]*.5566.com@/home/html/www.5566.com@g' test.txt
```

注意 Sed 是完全支持正则表达式的，在正则表达式里，`[^.]` 表示为非 . 的所有字符，换成 `[^/]` 也可。另外，@ 是 Sed 的分隔符，我们也可以用其他符号，比如 /，但是如果要用到 / 的话就得 \ 了，所以经常用的是 @。

4. 在配置 .conf 文件时，经常要为相邻的几行添加 # 号以注释掉

例如，我们要将 test.txt 文件中的 31~36 行加上 # 号，这该如何实现呢？

在 Vim 中，我们可以执行如下代码：

```
:31,36 s/^/#/
```

而用 Sed 的话则执行起来更方便，如下所示：

```
sed -i '31,36s/^/#/' test.txt
```

反之，如果要将 31~36 行带 # 号的全删除，用 Sed 该如何实现呢？方法如下：

```
sed -i '31,36s/^#//' test.txt
```

许多人习惯在这个方法后面带个 g，事实上，如果没有 g，则表示从行的左端开始匹配，每一行第一个与之匹配的会被换掉；如果有 g，则表示每一行所有与之匹配的都会被换掉。

5. 利用 Sed 很方便地分析日志

利用 Sed 还可以很方便地分析日志。例如，在以下的 secure 日志文件中，我想用 Sed 抓取 12:48:48 至 12:48:55 的日志。

```
Apr 17 05:01:20 localhost sshd[16375]: pam_unix(sshd:auth): authentication failure; logname = uid=0 euid=0 tty=ssh ruser= rhost=222.186.37.226 user=root
```

```
Apr 17 05:01:22 localhost sshd[16375]: Failed password for root from 222.186.37.226 port 60700 ssh2
```

```
Apr 17 05:01:22 localhost sshd[16376]: Received disconnect from 222.186.37.226: 11: Bye Bye
```

```
Apr 17 05:01:22 localhost sshd[16377]: pam_unix(sshd:auth): authentication failure; logname = uid=0 euid=0 tty=ssh ruser= rhost=222.186.37.226 user=root
```

```
Apr 17 05:01:24 localhost sshd[16377]: Failed password for root from 222.186.37.226 port
```



```

60933 ssh2
Apr 17 05:01:24 localhost sshd[16378]: Received disconnect from 222.186.37.226: 11: Bye Bye
Apr 17 05:01:24 localhost sshd[16379]: pam_unix(sshd:auth): authentication failure; logname =
uid=0 euid=0 tty=ssh ruser= rhost=222.186.37.226 user=root
Apr 17 05:01:26 localhost sshd[16379]: Failed password for root from 222.186.37.226 port
32944 ssh2
Apr 17 05:01:26 localhost sshd[16380]: Received disconnect from 222.186.37.226: 11: Bye Bye
Apr 17 05:01:27 localhost sshd[16381]: pam_unix(sshd:auth): authentication failure; logname =
uid=0 euid=0 tty=ssh ruser= rhost=222.186.37.226 user=root
Apr 17 05:01:29 localhost sshd[16381]: Failed password for root from 222.186.37.226 port
33174 ssh2
Apr 17 05:01:29 localhost sshd[16382]: Received disconnect from 222.186.37.226: 11: Bye Bye
Apr 17 05:01:29 localhost sshd[16383]: pam_unix(sshd:auth): authentication failure; logname =
uid=0 euid=0 tty=ssh ruser= rhost=222.186.37.226 user=root
Apr 17 05:01:31 localhost sshd[16383]: Failed password for root from 222.186.37.226 port
33474 ssh2
Apr 17 05:01:31 localhost sshd[16384]: Received disconnect from 222.186.37.226: 11: Bye Bye
Apr 17 05:01:32 localhost sshd[16385]: pam_unix(sshd:auth): authentication failure; logname =
uid=0 euid=0 tty=ssh ruser= rhost=222.186.37.226 user=root

```

我们可以用 `tail` 看到下面的日志最终时间是 05:01:32，那么如何用 `Sed` 抓取呢？利用 `Sed` 成功截取日志命令，如下所示：

```

cat /var/log/secure | sed -n '/12:48:48/,/12:48:55/p'
Apr 23 12:48:48 localhost sshd[20570]: Accepted password for root from 220.249.72.138 port
27177 ssh2
Apr 23 12:48:48 localhost sshd[20570]: pam_unix(sshd:session): session opened for user root by
(uid=0)
Apr 23 12:48:55 localhost sshd[20601]: Accepted password for root from 220.249.72.138 port
59754 ssh2

```

`Sed` 的用法还有许多，这就靠我们大家在工作中归纳总结了。有兴趣的朋友还可以多了解 `Awk` 的用法，我们在工作中要频繁地分析日志文件，`Awk + Sed` 是比较好的选择。当然了，如果大家精通 `Perl` 语言，那就可以更方便地工作了。

5.3 基础正则表达式

首先要记住的是：正则表达式与通配符不一样，它们表示的含义并不相同！

正则表达式只是一种表示法，只要工具支持这种表示法，那么该工具就可以处理正则表达式的字符串。`Vim`、`grep`、`Awk`、`Sed` 都支持正则表达式，也正是因为它们支持正则表达式，所以才显得很强大。我学习正则表达式的方法是先学习实例，掌握基础实例后理论基本也就到位了。所以这里我会举一些关于 `grep` 配合正则表达式的实例来说明正则表达式的强大之处。先来看看 `grep` 工具。

以前介绍过，`grep` 工具格式如下：

```
grep -[acinv] '搜索内容串' filename
```

其中：

□ `-a`：表示以文本文件方式搜索。

- -c: 表示计算找到符合行的次数。
- -i: 表示忽略大小写。
- -n: 表示顺便输出行号。
- -v: 表示反向选择, 即找到没有搜索字符串的行。

另外, 搜索内容串可以是正则表达式。

下面举例说明之:

1) 搜索有 the 的行, 并输出行号, 如下所示:

```
#grep -n 'the' regular_express.txt
```

搜索没有 the 的行, 并输出行号, 如下所示:

```
#grep -nv 'the' regular_express.txt
```

2) 利用 [] 搜索集合字符。

[] 表示其中的某一个字符, 例如 [ade] 表示 a、d 或 e。

```
#grep -n 't[ae]st' regular_express.txt
```

```
8:I can't finish the test.
```

```
9:Oh! the soup taste good!
```

可以用 ^ 符号做 [] 内的前缀, 表示除 [] 内的字符之外的字符。

比如, 要搜索 oo 前没有 g 的字符串所在的行, 就可以使用 '[^g]oo' 作为搜索字符串, 如下所示:

```
#grep -n '[^g]oo' regular_express.txt
```

```
2:apple is my favorite food.
```

```
3:Football game is not use feet only.
```

[] 内也可以用范围来表示, 比如 [a-z] 表示 26 个小写字母, [0-9] 表示 0~9 的数字, [A-Z] 则表示 26 个大写字母。[a-zA-Z0-9] 表示所有数字与英文字符。当然也可以配合 ^ 来排除字符。

搜索包含数字的行, 如下所示:

```
#grep -n '[0-9]' regular_express.txt
```

```
5:However ,this dress is about $3183 dollars.
```

```
15:You are the best is menu you are the no.1.
```

3) 行首字符 ^ 与行尾字符 \$。

符号 ^ 表示行的开头, \$ 表示行的结尾 (不是字符, 是位置), 那么 '^\$' 则表示空, 因为只有行首和行尾。这里的符号 ^ 与 [] 里面所使用的 ^ 意义不同, 它表示的是符号 ^ 后面的串是在行的开头。比如搜索 the 在开头的行, 如下所示:

```
#grep -n '^the' regular_express.txt
```

```
12:the symbol '*' is represented as star.
```

4) 搜索以小写字母开头的行。

命令如下所示:

```
#grep -n '^[a-z]' regular_express.txt
```

```
2:apple is my favorite food.
```

```

4:this dress doesn't fit me.
10:motorcycle is cheap than car.
12:the symbol '*' is represented as star.
18:google is the best tools for search keyword.
19:gooogle yes!
20:go! go! Let's go.
woody@xiaoc: ~/tmp $

```

5) 搜索开头不是英文字母的行。

命令如下所示:

```

grep -n '^[^a-zA-Z]' regular_express.txt
1:"Open Source" is a good mechanism to develop programs.
21:#I am VBird
woody@xiaoc: ~/tmp $

```

\$ 表示它前面的串是在行的结尾, 比如, '\.\$' 表示点(.)在一行的结尾。
搜索末尾是点(.)的行, 如下所示:

```
#grep -n '\.$' regular_express.txt
```

点(.)是正则表达式的特殊符号, 所以要用 \ 转义, 结果如下所示:

```

1:"Open Source" is a good mechanism to develop programs.
2:apple is my favorite food.
3:Football game is not use feet only.
4:this dress doesn't fit me.
5:However ,this dress is about $3183 dollars.
6:GNU is free air not free beer.
...

```

6) 注意在 MS 系统下生成的文本文件, 换行时会加上一个 ^M 字符, 所以最后的字符会是隐藏的 ^M, 在处理 Windows 下面的文本时要特别注意!

可以用 `cat dos_file | tr -d '\r' > unix_file` 来删除 ^M 符号。那么 '^\$' 就表示只有行首、行尾的空行了!

搜索空行的命令如下所示:

```

#grep -n '^$' regular_express.txt
22:
23:
woody@xiaoc: ~/tmp $

```

7) 搜索非空行的命令如下所示:

```

grep -vn '^$' regular_express.txt
1:"Open Source" is a good mechanism to develop programs.
2:apple is my favorite food.
3:Football game is not use feet only.
4:this dress doesn't fit me.
...

```

8) 正则中的重复字符 * 与任意一个字符点(.)。

在 bash 中 * 代表通配符, 用来表示任意个字符, 但是在正则表达式中, 其含义则不同, * 表示有 0 个或多个某个字符, 请注意区分一下。

例如, oo* 表示第一个 o 一定存在, 第二个 o 可以有一个或多个, 也可以没有, 因此代表至少一个 o, 点(.)代表一个任意字符, 必须存在。

在下面的例子中, g??d 可以用 'g.d' 表示, good、gxxd、gabd..... 都符合 g??d。

```
grep -n 'g.d' regular_express.txt
1:"Open Source" is a good mechanism to develop programs.
9:Oh! the soup taste good!
16:The world is the same with 'glad'.
woody@xiaoc: ~/tmp$
```

搜索有两个 o 以上的字符串, 如下所示:

```
grep -n 'ooo*' regular_express.txt

1:"Open Source" is a good mechanism to develop programs.
2:apple is my favorite food.
3:Football game is not use feet only.
9:Oh! the soup taste good!
18:google is the best tools for search keyword.
19:gooooooogle yes!
```

grep -n 'ooo*' regular_express.txt 表示前两个 o 一定存在, 第三个 o 可没有, 也可有多个。搜索以 g 开头和结尾, 中间是至少一个 o 的字符串, 即 gog、goog、gooug 等, 如下所示:

```
#grep -n 'goo* g' regular_express.txt
18:google is the best tools for search keyword.
19:gooooooogle yes!
```

9) 限定连续重复字符的范围时使用 {}。

符号.* 只能限制 0 个或多个字符, 如果要确切地限制字符的重复数量, 就要用 {范围} 这种方式。范围是数字, 用逗号(,)隔开。比如, 2, 5 表示 2~5 个, 2 表示 2 个, 2, 表示 2 到更多个。

注意 由于 {} 在 SHELL 中有特殊意义, 因此作为正则表达式用的时候要用 \ 转义一下。

搜索包含两个 o 的字符串的行, 如下所示:

```
#grep -n 'o\{2\}' regular_express.txt
1:"Open Source" is a good mechanism to develop programs.
2:apple is my favorite food.
3:Football game is not use feet only.
9:Oh! the soup taste good!
18:google is the best tools for search keyword.
19:gooooooogle yes!
```

搜索 g 后面跟 2~5 个 o, 再跟一个 g 的字符串的行, 如下所示:

```
# grep -n 'go\{2,5\}g' regular_express.txt
18:google is the best tools for search keyword.
```


搜索包含 g，且后面跟 2 个以上的 o，再跟 g 的行，如下所示：

```
#grep -n 'go\{2,\}g' regular_express.txt
18:google is the best tools for search keyword.
19:gooooooogle yes!
```

10) 注意，[] 中的符号 ^ 表示否定的意思，也可以把它放在 [] 中内容的后面。

'[^a-z\.\!^-]' 表示没有小写字母，没有点(.)，没有感叹号(!)，没有空格，没有 - 的串，注意 [] 里面有个小空格。另外 SHELL 里面的反向选择为 [!range]，而在正则表达式里则是 [^range]，希望大家也注意区分一下。

11) 扩展正则表达式 egrep。

扩展正则表达式是在基础正则表达式上添加了几个特殊符号构成的，它令某些操作更加方便。比如，我们要去除空白行和行首为 # 的行，会这样用：

```
#grep -v '^$' regular_express.txt | grep -v '^#'
"Open Source" is a good mechanism to develop programs.
apple is my favorite food.
Football game is not use feet only.
this dress doesn't fit me.
```

然而使用支持扩展正则表达式的 egrep 与扩展特殊符号 |，则会方便许多。

注意 grep 只支持基础表达式，而 egrep 支持扩展，其实 egrep 是 grep -E 的别名，因此 grep -E 支持扩展正则表达式。

用法如下所示：

```
#egrep -v '^$|^#' regular_express.txt
"Open Source" is a good mechanism to develop programs.
apple is my favorite food.
Football game is not use feet only.
this dress doesn't fit me.
```

这里的符号 | 表示或的关系。即满足 ^\$ 或 ^# 的字符串。

注意 egrep 也很好用，我在写一个 SHELL 脚本时就正好用到了这个。有时候，并不是只 grep -v 一个对象，像 egrep -v "192.168.1.101 | 102 | 104" 这种用法就很适合 egrep。

这里列出几个扩展特殊符号：

+: 与符号 . * 的作用类似，表示一个或多个重复字符。

?: 与符号 . * 的作用类似，表示 0 个或一个字符。

|: 表示或的关系，比如 'gd | good | dog' 表示有 gd、good 或 dog 的字符串。

()：将部分内容合成一个单元组。比如，要搜索 glad 或 good，可以采用 'g(la | oo)d' 这种方式。() 的好处是可以对小组使用 + ? * 等。

比如，要搜索以 A 和 C 开头结尾，中间有至少一个 (xyz) 的字符串，可以这样用：'A(xyz) + C'。

5.4 Linux 下强大的查找命令 find

这一节跟大家介绍一下 Linux 强大的查找命令 find。find 的强大就在于它完美地支持了正则表达式。由于 find 具有强大的功能，所以它的选项也很多，其中大部分选项都值得我们花时间来了解一下。即使系统中含有网络文件系统（NFS），find 命令在该文件系统中也同样有效，只要具有相应的权限。在运行一个非常消耗资源的 find 命令时，很多人都倾向于把它放在后台执行，因为遍历一个大的文件系统可能会花费很长的时间（这里是指 100GB 字节以上的文件系统）。

1. find 命令的格式

find 命令的一般形式如下：

```
find pathname -options [-print -exec -ok ...]
```

- **pathname**：是 find 命令所查找的目录路径。例如用符号 . 来表示当前目录，用 / 来表示系统根目录。
- **-print**：表示 find 命令将匹配的文件输出到标准输出中。
- **-exec**：表示 find 命令对匹配的文件执行该参数所给出的 SHELL 命令。相应命令的形式为 'command' || \;，注意 || 和符号 \; 之间的空格。
- **-ok**：它的作用和 -exec 相同，只不过是以一种更为安全的模式来执行该参数所给出的 SHELL 命令，即在执行每一个命令之前，都会给出提示，让用户来确定是否执行。

下面来看看 find 命令选项。

□ -name

按照文件名查找文件。

□ -perm

按照文件权限来查找文件。

□ -prune

使用这一选项可以使 find 命令不在当前指定的目录中查找，如果同时使用 -depth 选项，那么 -prune 将被 find 命令忽略。

□ -user

按照文件属主来查找文件。

□ -group

按照文件所属的组来查找文件。

□ -mtime -n +n

按照文件的更改时间来查找文件，-n 表示从此刻算起，文件的更改时间是在 n 天以内；+n 表示文件的更改时间是在 n 天以前。find 命令还有 -atime 和 -ctime 选项，它们和 -mtime 选项的时间规定类似。

□ -nogroup

查找无有效所属组的文件，即该文件所属的组在 /etc/groups 中不存在。

□ -nouser

查找无有效属主的文件，即该文件的属主在 /etc/passwd 中不存在。

□ `-newer file1 ! file2`

查找更改时间比文件 `file1` 新但比文件 `file2` 旧的文件。

□ `-type`

查找某一类型的文件，诸如：

`b`：表示块设备文件。

`d`：表示目录。

`c`：表示字符设备文件。

`p`：表示管道文件。

`l`：表示符号链接文件。

`f`：表示普通文件。

`-size n: [c]`：表示查找文件长度为 `n` 块的文件，带有 `c` 时表示文件长度以字节计。

`-depth`：表示在查找文件时，首先查找当前目录中的文件，然后再在其子目录中查找。

`-fstype`：表示查找位于某一类型文件系统上的文件，这些文件系统类型通常可以在配置文件 `/etc/fstab` 中找到，该配置文件中包含了本系统中有关文件系统的信息。

`-mount`：表示在查找文件时不跨越文件系统 `mount` 点。

`-follow`：表示如果 `find` 命令遇到符号链接文件，就跟踪至链接所指向的文件。

`-cpio`：表示对匹配的文件使用 `cpio` 命令，将这些文件备份到磁带设备中。

我们一般使用 `exec` 或 `ok` 来执行 SHELL 命令。

使用 `find` 时，只要把想要的操作写到一个文件里，就可以用 `exec` 来配合 `find` 查找，很方便。

在有些操作系统中只允许 `-exec` 选项执行诸如 `ls` 或 `ls -l` 这样的命令。大多数用户使用这一选项是为了查找旧文件并删除它们。建议在真正执行 `rm` 命令删除文件之前，最好先用 `ls` 命令查看一下，确认它们是所要删除的文件。

`exec` 选项后面跟随着所要执行的命令或脚本，然后是一对 `{}`、一个空格和一个 `\` 符号，最后是一个分号。为了使用 `exec` 选项，必须要同时使用 `print` 选项。如果验证一下 `find` 命令，你会发现该命令只输出从当前路径起的相对路径及文件名。

例如：为了用 `ls -l` 命令列出所匹配的文件，可以把 `ls -l` 命令放在 `find` 命令的 `-exec` 选项中，如下所示：

```
find . -type f -exec ls -l {} \;
-rw----- 1 root root 1024 06-20 20:34 ./rnd
-rw-r--r-- 1 root root 939 06-22 17:59 ./cat
-rw----- 1 root root 8022 06-22 19:47 ./viminfo
-rw-r--r-- 1 root root 939 06-22 17:59 ./list.txt
-rw-r--r-- 1 root root 3972 06-15 13:30 ./install.log.syslog
-rwxr-xr-x 1 root root 448 06-21 11:45 ./rsync.sh
-rw-r--r-- 1 root root 100 2007-01-06 ./cshrc
-rw-r--r-- 1 root root 36730 06-18 11:52 ./New_Text_Document.txt
-rw----- 1 root root 6 06-21 11:03 ./mysql_history
-rw-r--r-- 1 root root 789 06-18 23:24 ./ssh/known_hosts
-rw-r--r-- 1 root root 167 06-21 11:34 ./vimrc
-rw-r--r-- 1 root root 191 2007-01-06 ./bash_profile
-rw-r--r-- 1 root root 24 2007-01-06 ./bash_logout
```

```
-rw-r--r-- 1 root root 309 06-19 23:23 ./bashrc
-rw-r--r-- 1 root root 22772 06-15 13:30 ./install.log
-rw-r--r-- 1 root root 939 06-22 18:00 ./file
-rw----- 1 root root 1254 06-15 13:30 ./anaconda-ks.cfg
-rw-r--r-- 1 root root 129 2007-01-06 ./tcshrc
-rw----- 1 root root 13844 06-22 10:07 ./bash_history
```

在上面的例子中，find 命令匹配到了当前目录下的所有普通文件，并在 -exec 选项中使用 ls -l 命令将它们列出。

下面在 /logs 目录中查找更改时间在 5 日以前的文件并删除它们，如下所示：

```
find /logs -type f -mtime +5 -exec rm {} \;
```

注意 在 SHELL 中用任何方式删除文件之前，应当先查看相应的文件，一定要小心使用诸如 mv 或 rm 的命令。可以使用安全模式 -ok，它将在对每个匹配到的文件进行操作之前提示你。

在下面的例子中，find 命令在当前目录中查找所有文件名以 .log 结尾、更改时间在 5 日以上的文件，并删除它们，只不过在删除之前先给出提示。命令如下所示：

```
find . -name "*.log" -mtime +5 -ok rm {} \;
```

按 [y] 键删除文件，按 [n] 键不删除。

任何形式的命令都可以在 -exec 选项中使用。

在下面的例子中我们使用 grep 命令。先用 find 命令匹配所有文件名为 “passwd*” 的文件，例如 passwd、passwd.old、passwd.bak，然后执行 grep 命令看看在这些文件中是否存在一个 sam 用户。命令如下所示：

```
find /etc -name "passwd*" -exec grep "sam" {} \;
sam:x:501:501::/usr/sam:/bin/bash
```

2. find 命令的实例说明

下面是 find 命令的实例说明。

1) 查找当前用户主目录下的所有文件，命令如下：

```
find ~ -print
```

2) 让当前目录中的文件属主具有读、写权限，并且文件所属组的用户和其他用户具有读权限的文件，其实就是查找权限为 644 的文件，命令如下：

```
find . -type f -perm 644 -exec ls -l {} \;
```

3) 查找系统中所有文件长度为 0 的普通文件，并列出它们的完整路径，命令如下：

```
find / -type f -size 0 -exec ls -l {} \;
```

4) 查找 /var/logs 目录中更改时间在 7 日以前的普通文件，并在删除之前进行提示，命令如下所示：

```
find /var/logs -type f -mtime +7 -ok rm {} \;
```


5) 查找当前目录中所有属于 root 组的文件, 命令如下所示:

```
find . -group root -exec ls -l {} \;
-rw-r--r-- 1 root root 789 06-18 23:24 known_hosts
-rw-r--r-- 1 root root 789 06-18 23:24 ./ssh/known_hosts
-rw-r--r-- 1 root root 167 06-21 11:34 ./vimrc
-rw-r--r-- 1 root root 191 2007-01-06 ./bash_profile
-rw-r--r-- 1 root root 24 2007-01-06 ./bash_logout
-rw-r--r-- 1 root root 309 06-19 23:23 ./bashrc
-rw-r--r-- 1 root root 22772 06-15 13:30 ./install.log
-rw-r--r-- 1 root root 939 06-22 18:00 ./file
-rw----- 1 root root 1254 06-15 13:30 ./anaconda-ks.cfg
-rw-r--r-- 1 root root 129 2007-01-06 ./tcshrc
-rw----- 1 root root 13844 06-22 10:07 ./bash_history
```

6) find 命令将删除目录中访问时间在 7 日以内、且含有数字后缀的 admin.log 文件。

由于该命令只检查三位数字, 所以相应文件的后缀不要超过 999。我们先建几个 admin.log * 的文件, 然后再使用下面的命令删除之:

```
find . -name "admin.log[0-9]" -atime -7 -ok rm {} \;
```

7) 查找当前文件系统中的所有目录并排序, 命令如下:

```
find . -type d | sort
/data
/data/htdocs
/data/logs
/data/mysql
/data/mysql/3306
/data/mysql/3306/binlog
/data/mysql/3306/data
/data/mysql/3306/data/mysql
/data/mysql/3306/data/performance_schema
/data/mysql/3306/data/test
/data/mysql/3306/relaylog
```

与 SHELL 下自带的 ls 命令比较一下结果, 你会发现 find 命令可以列出当前目录下所有的目录, 我们可以根据需求来选择到底采用哪种方法。下面看看 SHELL 下查找目录的方法, 如下所示:

```
ls -lF | grep /\$
drwxrwxrwx 2 root root 4096 06-22 10:02 htdocs/
drwxr-xr-x 2 root root 4096 06-20 20:58 logs/
drwxr-xr-x 3 mysql mysql 4096 06-20 20:17 mysql/
```

8) 查找系统中所有的 rmt 磁带设备, 命令如下:

```
find /dev/rmt -print
```

3. 用 xargs 来配合 find 工作

在使用 find 命令的 -exec 选项处理匹配到的文件时, find 命令将所有匹配到的文件一起传递给 exec 执行。但有些系统对传递给 exec 的命令长度是有限制的, 这样, 在 find 命令运行几分钟之后,

就会出现溢出错误。错误信息通常是“参数列太长”或“参数列溢出”。这就是 xargs 命令的用处所在，特别是在与 find 命令一起使用时。

find 命令把匹配到的文件传递给 xargs 命令，而 xargs 命令每次只获取一部分文件而不是全部，不像 -exec 选项那样。这样它就可以先处理最先获取的那一部分文件，然后是下一批，并且如此继续下去。

在有些系统中，使用 -exec 选项会为处理每一个匹配到的文件而发起一个相应的进程，并非将匹配到的文件全部作为参数一次执行。这样，在有些情况下就会出现进程过多，系统性能下降的问题，因而效率不高。

而使用 xargs 命令则只有一个进程。另外，在使用 xargs 命令时，究竟是一次获取所有的参数，还是分批取得参数，包括每一次所获取参数的数目，都会根据该命令的选项及系统内核中相应的可调参数来确定。

来看看 xargs 命令是如何同 find 命令一起使用的，同时给出一些例子。

1) 下面的例子用来查找系统中的每一个普通文件，然后使用 xargs 命令来测试它们分别属于哪类文件。

```
#find . -type f -print | xargs file
./rnd: data
./cat: Non-ISO extended-ASCII text, with LF, NEL line terminators
./viminfo: ISO-8859 text
./list.txt: Non-ISO extended-ASCII text, with LF, NEL line terminators
./install.log.syslog: ASCII text
./rsync.sh: Bourne-Again shell script text executable
./cshrc: ASCII text
./New_Text_Document.txt: ISO-8859 English text, with very long lines, with CRLF line terminators
./admin.log2: empty
./mysql_history: ASCII text
./ssh/known_hosts: ASCII text, with very long lines
./vimrc: ASCII text
./bash_profile: ASCII English text
./bash_logout: ASCII text
./admin.log1: empty
./bashrc: ASCII text
./install.log: ASCII text
./file: Non-ISO extended-ASCII text, with LF, NEL line terminators
./anaconda-ks.cfg: ASCII English text
./tcshrc: ASCII text
./bash_history: ASCII text, with very long lines
```

2) 在当前目录下查找所有用户具有读、写和执行权限的文件，并收回相应的写权限，命令如下：

```
find . -perm -777 -print | xargs chmod o-w
```

然后我们用 ls -lsart 命令检查一下最后一句 find 命令是否生效，如下所示：

```
ls -lsart 1 2 3
0 -rwxrwxr-x 1 root root 0 06-22 20:09 1
0 -rwxrwxr-x 1 root root 0 06-22 20:09 2
```

```
0 -rwxrwxr-x 1 root root 0 06-22 20:09 3
[root@server andrewy0m]#
```

3) 用 `grep` 命令在所有的普通文件中搜索包含 `rmt` 这个词的文件。`find` 命令配合着 `exec` 和 `xargs` 使用, 可以使用户对所匹配到的文件执行几乎所有的命令, 如下所示:

```
find . -type f -print |xargs grep rmt
./install.log:Installing rmt -0.4b41-4.el5.x86_64
```

嫌麻烦的朋友也可以采用以下命令来查找:

```
grep rmt -rl .
```

`grep` 也有两个很厉害的参数, 一个是 `r`, 表示递归的意思; 另一个是 `l`, 表示列出来的意思。这条命令也会在当前目录下的所有目录和文件查找包含 `rmt` 字符的文件。

4. 更详细和强大的 `find` 实例

下面是 `find` 一些常用参数的例子, 先看看, 没必要死记硬背, 待要用的时候再找也不迟。

(1) 使用 `name` 选项

文件名 (`name`) 选项是 `find` 命令最常用的选项, 要么单独使用该选项, 要么和其他选项一起使用。

可以使用某种文件名模式来匹配文件, 记住要用引号将文件名模式引起来。

不管当前路径是什么, 如果想要在自己的根目录 `$HOME` 中查找文件名符合 `*.txt` 的文件, 使用 `~` 作为 '`pathname`' 参数, 波浪号 `~` 代表你的 `$HOME` 目录。

想要在当前目录及子目录中查找所有的 '`*.txt`' 文件, 可以如下命令:

```
find . -name "*.txt" -print
```

想要在当前目录及子目录中查找文件名以一个大写字母开头的文件, 可以用如下命令:

```
find . -name "[A-Z]*" -print
```

想要在 `/etc` 目录中查找文件名以 `host` 开头的文件, 可以用如下命令:

```
find /etc -name "host*" -print
```

想要查找 `$HOME` 目录中的文件, 可以用如下命令:

```
find ~ -name "*" -print
find . -print
```

要想让系统高负荷运行, 就从根目录开始查找所有的文件, 如下所示:

```
find / -name "*" -print
```

如果想在当前目录中查找文件名以两个小写字母开头, 跟着是两个数字, 最后是 `.txt` 的文件, 如下命令就能够返回名为 `ax37.txt` 的文件:

```
find . -name "[a-z][a-z][0-9][0-9].txt" -print
```

(2) 使用 `perm` 选项

文件权限模式即 `-perm` 选项, `find` 会按文件权限模式来查找文件, 不过最好使用八进制的权限

表示法。如果要在当前目录下查找文件权限位为 755 的文件，即文件属主可以读、写、执行，其他用户可以读、执行的文件，可以用如下命令：

```
find . -perm 755 -print
```

还有一种表达方法：在八进制数字前面要加一个横杠-，表示都匹配，如-007 就相当于 777，-006 相当于 666，如下所示：

```
find . -perm -006
./c
./b
./a
```

这里只是介绍一下这种用法，建议大家还是采用前面一种直观的方法。下面针对其中的部分参数进行说明：

- -perm mode：文件许可正好符合 mode。
- -perm + mode：文件许可部分符合 mode，如果是 +006，表示文件的某一项权限为 6，可以随便是哪一项，如果属主符合 6 权限，也可由 find 命令打印出来。
- -perm - mode：文件许可完全符合 mode，如果是-007，表示文件的所有权限都必须是 7，即 777。

(3) 忽略某个目录进行查找

如果在查找文件时希望忽略某个目录，因为你知道那个目录中没有你所要查找的文件，那么可以使用-prune 选项来指出需要忽略的目录。在使用-prune 选项时要当心，因为如果你同时使用了-depth 选项，那么-prune 选项就会被 find 命令忽略了。

如果希望在/home/andrewy 目录下查找文件，但不希望在/home/andrewy/tv 目录下查找，可以用：

```
find /home/andrewy/ -path "/home/andrewy/tv" -prune -o -type f -print
/home/andrewy/第九章.doc
/home/andrewy/1
/home/andrewy/3
/home/andrewy/.bash_profile
/home/andrewy/2
/home/andrewy/.bash_logout
/home/andrewy/.bashrc
/home/andrewy/test
/home/andrewy/file
```

(4) 使用 user 和 nouser 选项

按文件属主查找文件，如果要在 \$HOME 目录中查找文件属主为 sam 的文件，可以用如下命令：

```
find ~ -user sam -print
```

在/etc 目录下查找文件属主为 uucp 的文件，可以用如下命令：

```
find /etc -user uucp -print
```

为了查找属主账户已经被删除的文件，可以使用-nouser 选项，这样就能够找到那些属主在

/etc/passwd 文件中没有有效账户的文件了。在使用 -nouser 选项时，不必给出用户名，find 命令能够为你完成相应的工作。

例如，希望在 /home 目录下查找所有的这类文件，可以用如下命令：

```
find /home -nouser -print
```

(5) 使用 group 和 nogroup 选项

就像 user 和 nouser 选项一样，针对文件所属于的用户组，find 命令也具有同样的选项，为了在 /apps 目录下查找属于 gem 用户组的文件，可以用如下命令：

```
find /apps -group gem -print
```

要查找没有有效所属用户组的所有文件，可以使用 nogroup 选项。下面的 find 命令从文件系统的根目录处查找这样的文件：

```
find / -nogroup -print
```

(6) 按照更改时间或访问时间等查找文件

如果希望按照更改时间来查找文件，可以使用 mtime、atime 或 ctime 选项。如果系统突然没有可用空间了，很有可能是某一个文件的长度在此期间增长迅速造成的，这时就可以用 mtime 选项来查找这样的文件。

用减号 - 来限定更改时间在距今 n 日以内的文件，而用加号 + 来限定更改时间是 n 日以前的文件。如果希望在系统根目录下查找更改时间在 5 日以内的文件，可以用如下命令：

```
find / -mtime -5 -print
```

为了在 /var/adm 目录下查找更改时间在 3 日以前的文件，可以用如下命令：

```
find /var/adm -mtime +3 -print
```

(7) 查找比某个文件新或旧的文件

如果希望查找更改时间比某个文件新但比另一个文件旧的所有文件，可以使用 -newer 选项。它的一般形式如下：

```
newest_file_name ! oldest_file_name
```

查找更改时间比 temp 文件新的文件，可以用如下命令：

```
find . -newer temp -print
```

在进行系统维护时，我们经常遇到一种情况，就是发现磁盘在不停地发生写现象，这时候我们要把这个正在写的文件定位出来，可以选择建立一个文件，比如 test.txt，然后用以下命令找出这频繁写磁盘的文件（最好带上一些限制大小的参数来精确定位），如下所示：

```
find / -newer test.txt -print
```

(8) 使用 type 选项

在 /etc 目录下查找所有的目录，可以用如下命令：

```
find /etc -type d -print
```

在当前目录下查找除目录以外的所有类型的文件，可以用如下命令：

```
find . ! -type d -print
```

在/etc 目录下查找所有的符号链接文件，可以用如下命令：

```
find /etc -type l -print
```

(9) 使用 size 选项

可以按照文件长度来查找文件，这里所指的文件长度既可以用块（block）来计量，也可以用字节来计量。以字节计量文件长度的表达形式为 Nc；以块计量文件长度只用数字表示即可。在按照文件长度查找文件时，一般使用这种以字节表示的方式；在查看文件系统的大小，则用以块表示的方式，因为这时使用块来计量更容易转换。

在当前目录下查找文件长度大于 1M 字节的文件，命令如下所示：

```
find . -size +1000000c -print
```

在/home/apache 目录下查找文件长度恰好为 100 字节的文件，命令如下所示：

```
find /home/apache -size 100c -print
```

在当前目录下查找长度超过 10 块的文件（一块等于 512 字节），命令如下所示：

```
find . -size +10 -print
```

以上只是介绍一下 size 的各种表示方法，在工作中我们其实都是用 M 来进行比对的，最简单的方法通常也是最有效率的。

(10) 使用 depth 选项

在使用 find 命令时，我们可能希望先匹配所有的文件，再在子目录中查找。在 find 命令中使用 depth 选项就可以达到此目的。比如，当在使用 find 命令向磁带上备份文件系统时，我们希望首先备份所有的文件，其次再备份子目录中的文件。

在下面的例子中，find 命令从文件系统的根目录开始，查找一个名为 CON.FILE 的文件：

```
find / -name "CON.FILE" -depth -print
```

大家可以通过 Linux 下的 find 和 grep 这些强大的 SHELL 命令配合正则表达式，写出强大的 SHELL 脚本，以便快速有效的工作。

5.5 汇总 Linux/Unix 下的 bash 快捷键

近期在工作中发现，许多同事，尤其是我们部门的 PHP 开发同事，基本不用 Linux/Unix 下的快捷键，这严重影响了工作效率。所以我收集了一下 Centos5.5 及 FreeBSD8.1 下 bash 中命令行的快捷键。以下这些快捷键在 Centos5.5_x64、FreeBSD8.1_x64 下均可使用。另外，我在每条用法后加上了注释，帮助大家理解它们的作用。

□ Ctrl + A：切换到命令行开始。

这个操作跟 Home 实现的结果一样，但 Home 在某些 Unix 环境下无法使用，这时候便可以使用此组合键。在 Linux 下的 Vim 中，这个快捷组合键也是有效的，而且在 Windows 系列的许多文件编辑器里也有效。

□ Ctrl + E：切换到命令行末尾。

这个操作跟 End 实现的结果一样，但 End 键在某些 Unix 环境下无法使用，这时候便可以使用这个组合键。在 Linux 下的 Vim 中，这个快捷组合键也是有效的，而且在 Windows 的许多文件编辑器里也有效。

□ Ctrl + L: 清除屏幕内容，效果等同于 Clear。

□ Ctrl + U: 清除剪切光标之前的内容。

这个命令很有用，在 nslookup 里也是有效的。我有时看见同事一个字一个字地删除 SHELL 命令，十分崩溃，其实完全可以用一个 Ctrl + U 搞定。

□ Ctrl + K: 剪切清除光标之后的内容。

□ Ctrl + Y: 粘贴刚才所删除的字符。

此命令比较强悍，删除的字符有可能是几个字符串，但也极有可能是一行命令。比如说我们都出现过手误现象，这有点类似于 MS Office 中的 Ctrl + Z 这个组合键的作用。

□ Ctrl + R: 在历史命令中查找（这个非常好用，输入关键字就调出了以前的命令）。

这个命令我强烈推荐，当 history 比较多时，想找一个比较复杂的命令，直接用此快捷键，SHELL 便会自动查找并调用，方便极了。

□ Ctrl + C: 终止命令。

□ Ctrl + D: 退出当前终端。

□ Ctrl + Z: 转入后台运行。

不过，由 Ctrl + Z 转入后台运行的进程在当前用户退出后就会终止，所以不如用 nohup 命令 &，因为 nohup 命令的作用就是在用户退出之后让进程仍然继续运行，而现在有许多脚本和命令都要求在退出终端时仍然有效。

□ !!: 重复执行最后一条命令。

history 显示你执行过的所有编号 + 历史命令。可以使用其配合符号 ! 来执行某命令。

□ ↑ (Ctrl + P): 显示上一条命令。

□ ↓ (Ctrl + N): 显示下一条命令。

□ !\$: 显示系统最近的一条参数。

最后这个快捷键比较有用，比如我先用了 cat/etc/sysconfig/network-scripts/ifconfig-eth0，然后我想用 Vim 编辑。一般的做法是先用 ↑ 显示最后一条命令，接着用 Home 移动到命令最前面，删除 cat，然后再输入 Vim 命令。其实完全可以用 vim!\$ 来代替。

如果掌握以上用法，在 Linux/Unix 上工作基本上就非常有效率了。也许到最后，你会不经意地发现，弹指之间，许多复杂的 SHELL 指令就能很轻松地搞定，工作效率也越来越高了。

5.6 生产环境下的 SHELL 脚本分类

生产环境下的 SHELL 脚本作用还是挺多的，这里根据 4.1 节所介绍的日常工作中 SHELL 脚本的作用，将生产环境下的 SHELL 脚本分为备份类、监控类、统计类、开发类和自动化类。前面 3 类从字面意义上看比较容易理解，后面的我稍微解释一下：开发类脚本是用 SHELL 来配合 PHP 做一些非系统类的管理工作，比如 SVN 的发布程序等；而自动化类脚本则利用 SHELL 自动来替我们做一些繁琐的工作，比如自动生成及分配密码给开发组的用户或自动安装 LNMP 环境等。下面我就就这些分类举一些具体实例，便于大家理解。另外值得说明的一点是，这些实例都源自于我的线上

环境，大家拿过来稍微改动一下 IP 或备份目录基本上就可以使用了。

5.6.1 生产环境下的 SHELL 脚本备份类

俗话说得好，备份是救命的稻草。特别是重要的数据和代码，这些都是公司的重要资产，所以备份是必须的。备份能在我们执行了一些毁灭性的工作（比如不小心删除了数据）之后，进行恢复工作。我在实施项目时，发现有些有实力的公司在国内几个地方都有灾备机房，而且用的都是价格不菲的 EMC 高端存储。可能会有朋友要问：如果我们没有存储怎么办？这可以参考一下我们公司的备份策略，即：在执行本地备份的同时，让 SHELL 脚本自动上传数据到另一台 FTP 备份服务器中。这种备份策略成本比较小，无需存储。此双备策略的具体步骤如下所示：

首先，做好准备工作。我们先要安装一台 Centos5.5 的备份服务器，并安装 vsftpd 服务，稍微改动一下配置后启动。另外，关于 vsftpd 的备份目录，我们可以选择做 RAID1 或 RAID5 的分区或存储。没条件的朋友也可以像我们一样，用硬盘底座（HARD DRIVE DOCK）挂接一块 2TB 的 SATA2 硬盘（如果追求速度的话大家可以选择 e-SATA 插口），然后直接将其挂载到这台服务器的 /mnt 分区上。经过长时间的测试发现这也非常稳定，备份工作进行得很顺利。

vsftpd 服务的安装如下，Centos5.5 下自带的 yum 极为方便。

```
yum -y install vsftpd
service vsftpd start && chkconfig vsftpd on
```

关于 vsftpd 配置前面已经讲过，这里给出配置文件，我们可以通过组合使用如下命令直接得出 vsftpd.conf 中有效的文件内容，如下所示：

```
grep -v "^#" /etc/vsftpd/vsftpd.conf | grep -v '^$'
local_enable=YES
write_enable=YES
local_umask=022
dirmessage_enable=YES
xferlog_enable=YES
connect_from_port_20=YES
xferlog_std_format=YES
listen=YES
chroot_local_user=YES
pam_service_name=vsftpd
userlist_enable=YES
tcp_wrappers=YES
```

chroot_local_user = YES 这句话的作用我要重点强调一下。它的作用是限制所有本地用户在登录 vsftpd 服务器时只能在自己的家目录下，这是基于安全的考虑。在编写脚本的过程中也考虑到了这点，如果大家要移植此脚本到自己的工作环境中，不要忘了这句语法，不然的话异地备份极有可能失效。

另外，下例中的 CVS 目录备份脚本说明一个情况，我们在备份服务器（即 IP 为 192.168.4.45 的机器）上应该建立备份用户 CVS，并分配其密码，还应该将其家目录更改为备份目录，即 /data/backup/cvs-backup，这样的话更方便备份工作，以下备份脚本以此类推。FreeBSD8 的用户密码配置文件为 /etc/master.passwd，Centos5.5 为 /etc/passwd，两者是不一样的。由于这里用的 FTP 服务器

是 Centos5.5，所以我们只需要修改/etc/passwd 文件即可。

注意 下面的脚本源自于我的生产环境，并没有做任何变动，我对每个脚本运行的环境都进行了说明，大家也可以尝试将在 FreeBSD8 下运行的 SHELL 脚本移植到 Centos5.5 下面。注意 FreeBSD8 和 Centos5.5 下的 data 用法是不一样的，改编脚本时请注意！

1. CVS 目录的备份脚本

由于 CVS 服务器仅保留以前网站的代码和 CVS 日志，我们现不提交任何内容，只是以读为准，所以仅仅做目录级的备份即可。考虑到目录的备份不是太大，我就没有做轮询处理（即只备份某一周期的文件，比方说前 30 天），准备等备份文件过多时再考虑手动删除。应该在 vsftpd 服务器上建立相应的备份用户 cvs_user，脚本内容如下所示（此脚本在 FreeBSD8.1 x86_64 下已通过）：

```
#!/bin/sh
# CVS backup for freebsd8.1
# 2010-04-23

CVSDIR=/home/cvsroot/project
DATE='date +%Y-%m-%d'
OLDDATE='date -v -10d +%Y-%m-%d'

BACKDIR=/data/backup/cvs-backup
FILENAME=cvsbackup_'date +%Y-%m-%d'

if [ ! -d ${BACKDIR}/${DATE} ]; then
    mkdir ${BACKDIR}/${DATE}
fi

if [ -d ${BACKDIR}/${OLDDATE} ]; then
    rm -rf ${BACKDIR}/${OLDDATE}
fi

HOST=192.168.4.45
FTP_USERNAME=cvs_user
FTP_PASSWORD=cvs101

cd $CVSDIR
tar zcvf $FILENAME.tar.gz $CVSDIR

ftp -i -n -v << !
open ${HOST}
user ${FTP_USERNAME} ${FTP_PASSWORD}
bin
rm -rf ${OLDDATE}
mkdir ${DATE}
cd ${DATE}
mput *
bye
!
```

2. 版本控制软件 SVN 的代码库的备份脚本

版本控制软件 SVN 的重要性这里就不多说了，现在很多公司基本还是利用 SVN 作为提交代码集中管理的工具，所以做好其备份工作的重要性就不言而喻了。这里的轮询周期为 30 天一次，SHELL 会自动删除 30 天前的文件。应该在 vsftpd 服务器上建立相应的备份用户 svn_user，脚本内容如下（此脚本在 FreeBSD8 x86_64 下已通过）：

```
#!/bin/sh
# subversion backup for freebsd8.
# Created by ritto.zhao.
# 2010-04-23

SVNDIR=/data/svn
SVNADMIN=/usr/local/bin/svnadmin
DATE='date +%Y-%m-%d'
OLDDATE='date -v -30d +%Y-%m-%d'

BACKDIR=/data/backup/svn-backup
[ -d ${BACKDIR} ] || mkdir -p ${BACKDIR}
LogFile=${BACKDIR}/svnbak.log
[ -f ${LogFile} ] || touch ${LogFile}
mkdir ${BACKDIR}/${DATE}
fi

if [ -d ${BACKDIR}/${OLDDATE} ]; then
rm -rf ${BACKDIR}/${OLDDATE}
echo " " >> ${LogFile}
echo " " >> ${LogFile}
echo "-----" >> ${LogFile}
echo 'date +%Y-%m-%d %H:%M:%S' >> ${LogFile}
echo "-----" >> ${LogFile}
echo "* * * Subversion Backup Notification* * *" >> ${LogFile}
/usr/bin/printf "Host:    'hostname'\nAddress: ${IP}\nDate: ${DATE}\n"
>> ${LogFile}

for PROJECT in superbiiz ewizweb rest report ewiz
do
cd ${SVNDIR}
${SVNADMIN} hotcopy ${PROJECT} ${BACKDIR}/${DATE}/${PROJECT} --clean-logs
cd ${BACKDIR}/${DATE}
tar zcvf ${PROJECT}_svn_${DATE}.tar.gz ${PROJECT} > /dev/null
rm -rf ${PROJECT}

echo "Repository: ${PROJECT} backup done into ${BACKDIR}/${DATE}/ Successful!" >> ${LogFile}
echo "-----" >> ${LogFile}
echo "* * * Subversion Backup Notification* * *" >> ${LogFile}
/usr/bin/printf "Host:    'hostname'\nAddress: ${IP}\nDate:    ${DATE}\n"
>> ${LogFile}
```

```

for PROJECT in httpd web rest report test
do
    cd ${SVNDIR}
    ${SVNADMIN} hotcopy ${PROJECT} ${BACKDIR}/${DATE}/${PROJECT} --clean-logs

    cd ${BACKDIR}/${DATE}
    tar zcvf ${PROJECT}_svn_${DATE}.tar.gz ${PROJECT} > /dev/null
    rm -rf ${PROJECT}

    echo "Repository: ${PROJECT} backup done into ${BACKDIR}/${DATE}/ Successful!" >> ${LogFile}
    /bin/sleep 2
done

HOST=192.168.4.45
FTP_USERNAME=svn_user
FTP_PASSWORD=svn101

cd ${BACKDIR}/${DATE}

ftp -i -n -v << !
open ${HOST}
user ${FTP_USERNAME} ${FTP_PASSWORD}
bin
cd ${OLDDATE}
mdelete *
cd ..
rmdir ${OLDDATE}
mkdir ${DATE}
cd ${DATE}
mput *
bye
!

```

3. 公司内部局域网中重要服务器的相应资料备份

我们企业内部的 Wiki 服务器和 Apache 服务器上放了许多公司内部资料，所以也要注意对其进行备份。备份脚本跟前面的类似，30 天轮询一次，仍建议在 vsftpd 服务器上建立相应的备份用户，脚本内容如下所示（此脚本已在 FreeBSD8 x86_64 下测试通过）：

```

#!/bin/sh
#CVS backup for freebsd8
# Created by ritto.zhao.
# 2010-01-01.

DATE='date +%Y-%m-%d'
OLDDATE='date -v -30d +%Y-%m-%d'

BACKDIR=/data/backup/home
[ -d ${BACKDIR} ] || mkdir -p ${BACKDIR}

```

```

if [ ! -d ${BACKDIR}/${DATE} ]; then
    mkdir ${BACKDIR}/${DATE}
fi

if [ -d ${BACKDIR}/${OLDDATE} ]; then
    rm -rf ${BACKDIR}/${OLDDATE}
fi

cd /data
tar zcf ${BACKDIR}/${DATE}/web_wwwroot_backup_${DATE}.tar.gz wwwroot
/bin/sleep 2

cd /usr/local/www
tar zcf ${BACKDIR}/${DATE}/web_redmine_backup_${DATE}.tar.gz redmine-1.0
/bin/sleep 2

HOST=192.168.4.45
FTP_USERNAME=ewiz_svn
FTP_PASSWORD=ewiz_svn

cd ${BACKDIR}/${DATE}

ftp -i -n -v << !
open ${HOST}
user ${FTP_USERNAME} ${FTP_PASSWORD}
bin
cd ${OLDDATE}
mdelete *
cd ..
rmdir ${OLDDATE}
mkdir ${DATE}
cd ${DATE}
mput *
bye
!

```

4. MySQL 从数据库备份脚本

本着简单高效，不影响网站正常运营的原则，我们采用暂停从 MySQL 数据库的方式备份。这里要说明的是，本地的轮询周期为 20 天，vsftpd 的轮询周期为 60 天，备份后就算是用 gzip 压缩，MySQL 的数据库还是很大，大家可以根据实际需求来更改这个时间，注意不要让数据库撑爆你的备份服务器。由于是内部开发环境，密码设置得比较简单，如果要将其移植到自己的公网服务器上，还要在安全方面注意一下。另外附带说明一下，--opt 只是一个快捷选项，等同于同时添加--add-drop-tables --add-locking --create-option --disable-keys --extended-insert --lock-tables --quick --set-charset 选项。本选项能让 mysqldump 很快地导出数据，并且导出的数据可以很快导回。该选项默认是开启的，但可以用--skip-opt 禁用。注意，如果运行 mysqldump 没有指定--quick 或--opt 选项，则会将整个结果集放在内存中。如果导出的是大数据库的话可能会出现内存问题。脚本内容如下（此脚本在 Centos5.4 x86_64 下已通过）：


```
#!/bin/bash

USERNAME=mysqlbackup
PASSWORD=mysqlbackup

DATE='date +%Y-%m-%d'
OLDDATE='date +%Y-%m-%d -d '-20 days''
FTPOLDDATE='date +%Y-%m-%d -d '-60 days''

MYSQL=/usr/local/mysql/bin/mysql
MYSQLDUMP=/usr/local/mysql/bin/mysqldump
MYSQLADMIN=/usr/local/mysql/bin/mysqladmin
SOCKET=/tmp/mysql.sock

BACKDIR=/data/backup/db
[ -d ${BACKDIR} ] || mkdir -p ${BACKDIR}
[ -d ${BACKDIR}/${DATE} ] || mkdir ${BACKDIR}/${DATE}
[ ! -d ${BACKDIR}/${OLDDATE} ] || rm -rf ${BACKDIR}/${OLDDATE}

for DBNAME in mysql test report
do
    ${MYSQLDUMP} --opt -u ${USERNAME} -p ${PASSWORD} -S ${SOCKET} ${DBNAME} | gzip > ${BACK-
DIR}/${DATE}/${DBNAME}-backup-${DATE}.sql.gz
    echo "${DBNAME} has been backup successful"
    /bin/sleep 5
done

HOST=192.168.4.45
FTP_USERNAME=dbmysql
FTP_PASSWORD=dbmysql

cd ${BACKDIR}/${DATE}

ftp -i -n -v << !
open ${HOST}
user ${FTP_USERNAME} ${FTP_PASSWORD}
bin
cd ${FTPOLDDATE}
mdelete *
cd ..
rmdir ${FTPOLDDATE}
mkdir ${DATE}
cd ${DATE}
mput *
bye
!
```

5. mysqlhotcopy 热备份数据库脚本

mysqlhotcopy 是一个 Perl 脚本，最初由 Tim Bunce 编写并提供。它的主要实现原理是先锁住 (LOCK) 表，然后执行 FLUSH TABLES 动作，该正常关闭的表正常关闭，该进行 flush 同步的数据

也进行同步,然后通过执行系统级别的复制(cp等)命令,将需要备份的表或数据库的所有物理文件都复制到指定的备份集位置上。它主要用于备份 MyISAM 存储引擎的数据库。脚本内容如下(此脚本在 Centos5.5 x86_64 下已通过):

```
#!/usr/bin/env bash
#author:coralzd powered by www.freebsdssystem.org
#description: mysqlhotcopy backup script
DATE='date +%y%m%d.%H'
DATADIR="/data0/mysql/data" #mysql 数据库目录
BACKDIR_DAILY="/data0/local/mysqld/daily" #每天备份数据库的目录
BACKDIR_HOURLY="/data0/local/mysqld/hourly" #每小时备份数据库的目录
INTERVAL=" $1 "
TMPDIR="/var/tmp/" #临时目录
TMPDIRH="$TMPDIR"hourly #临时每小时数据库目录
TMPDIRD="$TMPDIR"daily #临时每小时备份数据库的目录
LOGDIR="/data0/log/dbbackup/" #备份数据输出日志路径
KEEPH_LOCAL=1
USER=hotcopy #mysqlhotcopy 用到的用户,必须要有 select,reload,lock_tables 权限
PASSWD=hotcopy # 密码
HOST='hostname -s'
MYVERSION="5.1"
mkdir -p $LOGDIR
case $INTERVAL in
hourly | HOURLY | Hourly | 1 )
echo "" && echo
" ~~~~~ "
echo "Performing HOURLY level backup -- 'date + $m - $d.%H:%M:%S'" && echo
""
echo "" && echo
" ~~~~~ "
>>"$LOGDIR"hourly.log
echo "Performing HOURLY level backup -- 'date + $m - $d.%H:%M:%S'"
>>"$LOGDIR"hourly.log
mkdir -p $TMPDIRH # 创建临时目录
mkdir -p $BACKDIR_HOURLY #创建每小时备份目录
/usr/local/mysql/bin/mysqlhotcopy --allowold test -u $USER -p $PASSWD
$TMPDIRH > /dev/null
echo "" && echo "" && echo
" ~~~~~ "
echo "Compressing -- 'date + $m - $d.%H:%M:%S'"
echo "Compressing -- 'date + $m - $d.%H:%M:%S'" >>"$LOGDIR"hourly.log
cd $TMPDIRH #进入临时目录
find . -maxdepth 1 -type d -user mysql -exec tar czf {} -"$DATE".tar.gz '{}' \; # 查找是 mysql
用户的数据,然后进行打包
find . -maxdepth 1 -type d -user mysql -exec rm -rf {} \; # 查找并删除用户名为 mysql 的文件
echo ""
echo "Copying Local... -- 'date + $m - $d.%H:%M:%S'"
echo "Copying Local... -- 'date + $m - $d.%H:%M:%S'" >>"$LOGDIR"hourly.log
chattr -i $BACKDIR_HOURLY # 移除备份目录的保护属性
find $BACKDIR_HOURLY -type f -mtime +7 -exec rm {} \; #查找并删除在 7 天以外的备份数据库
```

```

mv * $BACKDIR_HOURLY #将打包好的文件移到每小时备份目录
chattr +i $BACKDIR_HOURLY #对备份目录加保护
echo ""
echo "Ending -- 'date + $m - $d.%H:%M:%S'" >> "$LOGDIR"hourly.log
exit 0
;;
daily | DAILY | Daily | 2)
echo "" && echo
" ~~~~~ "
echo "Performing DAILY level backup -- 'date + $m - $d.%H:%M:%S'" && echo
""
echo "" && echo
" ~~~~~ "
>> "$LOGDIR"daily.log
echo "Performing DAILY level backup -- 'date + $m - $d.%H:%M:%S'"
>> "$LOGDIR"daily.log
mkdir -p $TMPDIR #进入临时目录
mkdir -p $BACKDIR_DAILY #创建每天备份目录
/usr/local/mysql/bin/mysqlhotcopy --allowold test -u $USER -p $PASSWD
$TMPDIR > /dev/null
echo "" && echo
" ~~~~~ "
echo "Compressing -- 'date + $m - $d.%H:%M:%S'"
echo "Compressing -- 'date + $m - $d.%H:%M:%S'" >> "$LOGDIR"daily.log
cd $TMPDIR #进入临时目录
find . -maxdepth 1 -type d -user mysql -exec tar czf {} - "$DATE".tar.gz '{}' \; #查找是mysql 用
户的数据库,然后进行打包
find . -maxdepth 1 -type d -user mysql -exec rm -rf {} \; #查找备份数据库,并将在1天以外的备份删除
echo "Copying Local... -- 'date + $m - $d.%H:%M:%S'"
echo "Copying Local... -- 'date + $m - $d.%H:%M:%S'" >> "$LOGDIR"daily.log
chattr -i $BACKDIR_DAILY #移除备份目录的保护属性
find $BACKDIR_DAILY -type f -mtime + $KEEPD_LOCAL -exec rm -rf '{}' \; #查找备份数据库,并将在1
天以外的备份删除
mv * $BACKDIR_DAILY #将打包好的文件移到每小时备份目录
chattr +i $BACKDIR_DAILY #对备份目录加保护
echo "Ending -- 'date + $m - $d.%H:%M:%S'" >> "$LOGDIR"daily.log
exit 0
;;
* )
echo ""
echo
" ~~~~~ "
echo "Invalid Selection" 1>&2
exit 1
esac

```

5.6.2 生产环境下的开发类 SHELL 脚本

SHELL 作为系统管理脚本,在开发功能上一直有所欠缺,但我们可以将其作为 PHP 或 Python 开发语言的辅助语言,配合作开发工具,提高我们的工作效率。

1. 配合 PHP 作为 SVN 的发布程序

公司的 SVN 发布程序是这样的工作流程：把将要发布的程序打上 publish please:，后面接上 SVN 日志（开发人员提交的代码日志，这比较重要，是后来更改代码的条件依据），然后 PHP 程序会检索出这些打上标的代码，并发布到线上环境中。这样一来就带来了一个问题：在按照原先的程序发布到线上后，会将 publish please 后面的开发人员写的 SVN log 清除，给开发造成了不便。我在接手此工作后将此 SHELL 脚本重写，解决了这个 Bug。由于此脚本牵涉的内容比较多，功能也比较繁琐，所以这里只给大家演示一下，并不要求掌握。实现功能的代码如下所示（此脚本在 FreeBSD8 x86_64 下已通过）：

```
#!/bin/sh
# SVN PUBLISH TOOL
# 2010-07-23
SVN=/usr/local/bin/svn

cd /var/www/html/svn_log/files/svn/
svn up --username web --password web101
cat /var/www/html/svn_log/files/svntest/file.log |svn log $2 |head -n5 |sed -n "/publish/p"
> svnlog
svnlog_change='sed -i "s@publish please@already published@" svnlog'
cat /var/www/html/svn_log/files/svntest/file.log |awk -F ',' '{ print "/usr/local/bin/svn
propset --username web --password web101 -r" $1" --revprop svn:log " $svnlog_change" " $2 }' | /
bin/sh

[ ! -d /var/www/html/svn_log/files/svntest ] || mv /var/www/html/svn_log/files/svntest /var/
www/html
/old-svntest/html - ${COMMIT}
mkdir /var/www/html/svn_log/files/svntest
chmod -R 777 /var/www/html/svn_log/files/svntest
chown -R www:www /var/www/html/svn_log/files/svntest
cd /var/www/html/svn_log/files/svn/
svn up --username web --password web101
```

此脚本在线上已稳定运行半年，基本能实现我们的要求。美中不足的是，我们发现还是存在着一个 Bug，即如果有大量脚本要同时提交的话，此脚本就失效了，这个 Bug 目前也只能在后续版本中改进了。

2. 利用 SHELL 安全方便地删除指定的 zone

我维护的 DNS 服务器主要有 3 个，即一主一从一备。由于公司的架构采用了 CDN 方案，所以 named.conf 针对“okspace.com”的出现位置就有 3 处，即电信、网通及其他，加上 3 个服务器，每次手动用 Vim 删除 okspace.com 时就必须修改 9 处，维护起来很麻烦。更不爽的是，有些 zone 经常需要删除，特别麻烦。所以我写了个 SHELL 脚本以减轻自己的工作量，且达到安全删除的目的。

此脚本中变量 domain 中的文件内容可以自己定义，鉴于生产环境下 bind 都是源码安装的，所以就以 named.conf 文件为主了。代码虽然短小，但操作起来确实很方便，后期我想跟 Python 或 PHP 结合起来使用，做成图形化界面，并加入了更多功能，使其更完善。代码如下（此脚本在 CentOS5.2 x86_64 下已通过）：


```
#!/bin/bash
domain='zone\ "okspace.cn"'
if [ -e /var/named/chroot/etc/named.conf ];then
sed -i "/ $domain/,/;/d" /var/named/chroot/etc/named.conf
else
sed -i "/ $domain/,/;/d" /var/named/chroot/var/named/named.rfc1912.zones
fi
```

在实际工作中我们了解了 SHELL 脚本强大的系统管理功能，也意识到了它在开发上的先天不足，这里我建议大家有时间的话可以关注和学习一门开发语言，PHP 或 Python 均可。大家可以根据自己的兴趣和职业规划，有重点地选择学习。

5.6.3 生产环境下的统计类 SHELL 脚本

统计工作一直是 SHELL 脚本的强项，我们完全可以利用 Sed、Awk 再加上正则表达式，写出强大的统计脚本来分析我们的系统日志、安全日志及服务器应用日志等。

1. Nginx 负载均衡器日志汇总脚本

以下脚本是用来分析 Nginx 负载均衡器的日志，作为 Awstats 的补充，它可以快速得出排名最前的网站和 IP 等。脚本内容如下所示（此脚本在 Centos5.5 x86_64 下已测试通过）：

```
#!/bin/bash

if [ $# -eq 0 ]; then
    echo "Error: please specify logfile."
    exit 0
else
    LOG = $1
fi

if [ ! -f $1 ]; then
    echo "Sorry, sir, I can't find this apache log file, pls try again!"
    exit 0
fi

#####
echo "Most of the ip:"
echo "-----"
awk '{ print $1 }' $LOG | sort | uniq -c | sort -nr | head -10
echo
echo
#####
echo "Most of the time:"
echo "-----"
awk '{ print $4 }' $LOG | cut -c 14-18 | sort | uniq -c | sort -nr | head -10
echo
echo
#####
echo "Most of the page:"
echo "-----"
```

```

awk '{print $11}' $LOG | sed 's/^.* \(.cn* \)\\"/\1/g' | sort | uniq -c | sort -rn | head -10
echo
echo
#####
echo "Most of the time / Most of the ip:"
echo "-----"
awk '{ print $4 }' $LOG | cut -c 14-18 | sort -n | uniq -c | sort -nr | head -10 > timelog

for i in `awk '{ print $2 }' timelog`
do
    num=`grep $i timelog | awk '{ print $1 }'`
    echo " $i $num"
    ip=`grep $i $LOG | awk '{ print $1 }' | sort -n | uniq -c | sort -nr | head -10`
    echo " $ip"
    echo
done
rm -f timelog

```

2. Nginx_web 服务器日志切割脚本

为什么我们要对日志进行切割呢？这是因为在生产环境下，我们的 Web 服务器是 365 × 24 小时不间断地提供服务的，它的日志也是不间断的产生的。而我们分析它的日志时，可能需要以天为单位来查看，所以考虑以天为单位来切割，可以利用 Crontab 来完成这项工作，切割后的日志可以用 Awstats 来分析。脚本内容如下所示（此脚本在 Centos5.5 x86_64 下已通过）：

```

#!/bin/bash
# The Nginx logs path
logs_path="/data0/logs"
logs_dir=${logs_path}/${date -d "yesterday" +"%Y"}/${date -d "yesterday" +"%m"}
logs_file=${date -d "yesterday" +"%Y%m%d"}
mkdir -p /data0/backuplogs/${date -d "yesterday" +"%Y"}/${date -d "yesterday" +"%m"}
tar -czf ${logs_path}/${logs_file}.tar.gz ${logs_path}/* .log
rm -rf ${logs_path}/* .log
mv ${logs_path}/${logs_file}.tar.gz /data0/backuplogs/${date -d "yesterday" +"%Y"}/${date -d "yesterday" +"%m"}
/usr/local/nginx/sbin/nginx -s reload
for oldfiles in `find /data0/backuplogs/${date -d "30 days ago" +"%Y"}/${date -d "30 days ago" +"%m"} -type f -mtime +30`
do
    rm -f $oldfiles
done

```

for oldfiles in 'find/data0/backuplogs/\${date -d"30 days ago" +"%Y"}/\${date -d"30 days ago" +"%m"} -type f -mtime +30' 这句代码的作用是自动删除一个月前的打包日志文件。Nginx 处理日志的脚本比较简单，大家没必要像我一样写得这么复杂，还可以简化一些。

3. Awstats Log 处理日志脚本

Awstats 安装路径要跟脚本中 awstats 的路径相对应，脚本如下所示（此脚本在 Centos5.5 x86_64 下已通过）：

```
#!/bin/sh
#Created by coralzd,build 20091226

today='date +%Y%m%d' --date='yesterday'
tday='date +%d' --date='yesterday'
year='date +%Y'
month='date +%m'

for dmain1 in "blog" "kx"; do # for 循环依次执行将相关应用按流程执行
    cd /acclogs/${dmain1}.example.com #该目录是可以随意变动的,但是要跟 awstats 各个站点的配置文档里的
    logfile 保持一致;大家根据自身业务作相应调整,需注意的是,要在 Web 通过 scp 命令或者 nfs 将产生的日志文件发送
    到/acclogs/相应的目录. 比如:blog web(只有一台,多台参照下 for 循环写的格式来做)的日志(其名字格式是年月日
    .1,比如 20110428.1)发送到/acclogs/blog.example.com/里
    chmod 777 *
    mv $today.1 access.log
    chmod 777 *
    /data0/www/wwwroot/awstats/tools/awstats_buildstaticpages.pl -update -config=${dmain1}.
    example.com -lang=cn -databasebreak=day -day=$tday -builddate=$today -dir=/var/web-
    hosts/htdocs/www -awstatsprog=/usr/local/webserver/awstats/wwwroot/cgi-bin/awstats.pl
    rm -rf /acclogs/${dmain1}.example.com/*
    done
    for dmain in "www" "edu" "finance" "health" "home" "club" "elec"; do # 还是用 for 循环依次将相关应用
    按流程执行
        cd /acclogs/${dmain}.example.com #该目录是可以随意变动的,但是要跟 awstats 各个站点的配置文档里的
        logfile 保持一致;大家根据自身业务作相应调整,需注意的是要在 Web 通过 scp 命令或者 nfs 将产生的日志文件发送到/
        acclogs/相应的目录. 比如:blog web(只有一台,多台参照下 for 循环写的格式来做)的日志(其名字格式是年月日.1,比
        如 20110428.1)发送到/acclogs/blog.example.com/里
        chmod 777 *
        sort -m -t " " -k 4 -o access.log $today.1 $today.2 $today.3 # 将 3 台 Web 的相同应用的日志合并
        为一个日志文件
        chmod 777 *
        mkdir -p /data0/www/wwwroot/www/${dmain}.example.com/$year/$month #应用当年当月目录
        /data0/www/wwwroot/awstats/tools/awstats_buildstaticpages.pl -update -config=club.exam-
        ple.com -lang=cn -databasebreak=day -builddate=$today -day=$tday -dir=/data0/www/wwwro-
        ot/www/club.example.com/$year/$month -awstatsprog=/data0/www/wwwroot/awstats/wwwroot/cgi-
        bin/awstats.pl
        rm -rf /acclogs/${dmain}.example.com/* #删除应用目录之前发送的日志
    done
```

4. 测试局域网内主机依然存活的小脚本

我们在对局域网的网络情况进行维护时,经常有这样的情况,需要收集网络中存活的 IP,这个时候可以写一个 SHELL 脚本,自动收集某一网段的 IP。现在的 IT 技术型公司一般都比较,网络工程师一般会规划几个 VLAN,即几个网段,我们可以多设几个变量参数来自动收集所有网段的 IP。脚本如下所示(此脚本在 RHEL5.1 x86_64、Centos5.5 x86_64 下已通过):

```
#!/bin/bash
#2008-10-01
for n in {100..200}; do
    host=192.168.4.$n
```

```

ping -c2 $host &>/dev/null
if [ $? = 0 ]; then
    echo "$host is UP"
    echo "$host" >> /root/alive.txt
else
    echo "$host is DOWN"
fi
done

```

程序会自动记录存活的 IP，我们用 cat 命令来查看 /root/alive.txt 文件，如下所示：

```

cat /root/alive.txt
192.168.4.100
192.168.4.101
192.168.4.102
192.168.4.108
192.168.4.109
192.168.4.111
192.168.4.113
192.168.4.121
192.168.4.123
192.168.4.125
192.168.4.133
192.168.4.138
192.168.4.142
192.168.4.143

```

脚本虽然短小，但却非常实用，免得又要到 Windows XP 下去下载局域网检测工具。我们平时在工作中也应该注意多收集多写一些这样的脚本，达到简化我们工作的目的。

5.6.4 生产环境下的监控类 SHELL 脚本

在生产环境下，服务器的稳定情况会直接影响公司的生意和信誉，可见其有多重要。所以，我们需要即时掌握服务器的状态，一般我们会在机房部署 Nagios 作为监控程序，然后用 SHELL 作为补充脚本，实时监控我们的服务器。

1. 自动监控 ADSL 状态并重启的脚本

故障分析：公司的办公环境中以一台 Ubuntu 8.04 作 NAT 路由器，但它的 ADSL 爱掉线，一掉的话网关的 Gateway 就没有了，直接影响就是所有同事都上不了网。所以我针对这种情况写了一个监控脚本，测试了很长时间，效果不错。执行脚本方法如下所示：

```
nohup sh route.sh &
```

注意前面要用上 nohup，这样可避免 root 用户 logout 时此脚本退出。脚本代码如下（此脚本在 Ubuntu8.04 i386 下已通过）：

```

#!/bin/bash
#Created By Andrew.Yu
while :
do
    if route | tail -1 | grep "0.0.0.0"

```



```

then
&>/dev/null
else
    adsl-stop
    adsl-start
fi
sleep 10
done

```

2. Nginx 负载均衡服务器上监控 Nginx 进程的脚本

由于我们的电子商务前端的 Load Balance 用到了 Nginx + Keepalived 架构，而 Keepalived 无法进行 Nginx 服务的实时切换，所以这里用了一个监控脚本 `nginx_pid.sh`，每隔 5 秒钟就监控一次 Nginx 的运行状态，如果发现有问题的就关闭本机的 Keepalived 程序，让 VIP 切换到从 Nginx 负载均衡器上。在对线上环境进行操作的时候，我人为地重启了主 Master 的 Nginx 机器，从 Nginx 机器在很短的时间内就接管了 VIP 地址，即网站的实际内网地址（此内网地址能过防火墙映射为公网 IP），进一步证实了此脚本的有效性。脚本内容如下（此脚本已在 Centos5.5 x86_64 下测试通过）：

```

#!/bin/bash
#Created By Andrew.Yu
while :
do
    nginxpid='ps -C nginx --no-header |wc -l'
    if [ $nginxpid -eq 0 ];then
        ulimit -SHn 65535
        /usr/local/nginx/sbin/nginx
    sleep 5
    nginxpid='ps -C nginx --no-header |wc -l'
    if [ $nginxpid -eq 0 ];then
        /etc/init.d/keepalived stop
    fi
fi
sleep 5
done

```

3. 系统文件打开数监测脚本

这个脚本比较方便，可用来查看 Nginx 进程下最大文件打开数。脚本代码如下（此脚本在 Centos5.5 x86_64 下已通过）：

```

#!/bin/bash
for pid in `ps aux |grep nginx |grep -v grep |awk '{print $2}'`
do
    cat /proc/${pid}/limits |grep 'Max open files'
done

```

脚本的运行结果如下所示：

Max open files	1024	1024	files
Max open files	65535	65535	files
Max open files	65535	65535	files
Max open files	65535	65535	files

Max open files	65535	65535	files
Max open files	65535	65535	files
Max open files	65535	65535	files
Max open files	65535	65535	files
Max open files	65535	65535	files
Max open files	65535	65535	files

4. Web 网站即时监控与报警程序

虽然现在规模较大的公司可以购买 AlertBot 即时监控服务, 并配合 Nagios 来监控网站, 但很多时候客户的公司可能没这个需求, 所以我们可以写一个 SHELL 程序监控我们的脚本, 网站出故障后就向我们的邮箱或手机发送信息报警。如果飞信接口不是太稳定的话, 可以向 139 的邮箱发报警邮件, 它会自动绑定到我们的移动手机上。脚本代码如下 (此脚本在 Centos5.5 x86_64 下已通过):

```
#!/bin/sh
# Created by kerryhu
# MAIL:king_819@163.com
# BLOG:http://kerry.blog.51cto.com
LANG = C

#被监控服务器、端口列表
server_all_list = (
125.252.87.171:80 \
58.64.157.166:80 \
58.64.157.174:80 \
#219.87.10.1:80 \
)

telnum=152XXXXXXX
passwd=12345

date = $(date -d "today" +"%Y-%m-%d_%H:%M:%S")

#采用 HTTP POST 方式发送检测信息给接口程序 interface.php, 接口程序负责分析信息, 决定是否发送报警 MSN 消息、手机短信、电子邮件
send_msg_to_interface()
{
    if [[ $2 = "0" ]] || [[ $2 = "2" ]]; then
        #开始发送警报邮件, 135XXXXXXXX@139.com 既是发送方也是接收方
        sendEmail -f 135XXXXXXXX@139.com -t 135XXXXXXXX@139.com -s smtp.139.com -u "from
http_monitor" -xu 135XXXXXXXX@139.com -xp james -o message -charset=utf-8 -m $1
        #发送 MSN 警报消息( 如果不需要 MSN 警报可以注释这行)
        # curl -m 600 -d menu=http -d date=$date -d ip=$server_ip -d port=$server_port
        -d status=$status http://127.0.0.1/monitor/interface.php
    fi;
}

server_all_len = ${#server_all_list[*]}
i=0
while [ $i -lt $server_all_len ]
```

```

do
server_ip=$(echo ${server_all_list[$i]} | awk -F ':' '{print $1}')
server_port=$(echo ${server_all_list[$i]} | awk -F ':' '{print $2}')
server_message=""
if curl -m 10 -G http://${server_all_list[$i]}/ > /dev/null 2>&1
then
#status: 0,http down 1,http ok 2,http down but ping ok
status=1
echo "服务器 ${server_ip},端口 ${server_port}能够正常访问!";
server_message="服务器 ${server_ip},端口 ${server_port}能够正常访问!";
else
if curl -m 30 -G http://${server_all_list[$i]}/ > /dev/null 2>&1
then
status=1
echo "服务器 ${server_ip},端口 ${server_port}能够正常访问!"
server_message="服务器 ${server_ip},端口 ${server_port}能够正常访问!";
else
if ping -c 1 $server_ip > /dev/null 2>&1
then
status=2
echo "服务器 ${server_ip},端口 ${server_port}无法访问,但是能够 Ping 通!";
server_message="服务器 ${server_ip},端口 ${server_port}无法访问,但是能够
Ping 通!";
else
status=0
echo "服务器 ${server_ip},端口 ${server_port}无法访问,并且无法 Ping 通!";
server_message="服务器 ${server_ip},端口 ${server_port}无法访问,并且无法
Ping 通!";
fi
fi
fi
send_msg_to_interface "${server_message}" "${status}";
let i++
done

```

5.6.5 生产环境下的自动化类 SHELL 脚本

1. 批量生成账户脚本

在内网开发环境中,有时需要为开发组的同事批量生成账户,如果手动添加的话会非常麻烦,这个时候我们可以写一段 SHELL 脚本来自动完成这项工作。在首次登录时密码均是统一的,在移交给开发人使用时让他们自行更改。脚本代码如下(此脚本在 Centos5.5 x86_64 下已通过):

```

#!/bin/bash
#此脚本应用于开发环境下生成批量用户
for name in tom jerry joe jane andrewy brain
do
useradd $name
echo redhat |passwd --stdin $name
done

```

passwd --stdin 的作用是将前面的输入通过管道命令作为自己的输出，从而避免脚本交互，达到自动化的目的。由于 FreeBSD8 的许多命令跟 Centos5.5 不一样，以上脚本并不适合 FreeBSD8 系统，有兴趣的朋友可以自动研究。

2. 系统初始化脚本

此脚本用于新装 Linux 的相关配置工作，比如禁掉 iptable 和 SELinux 及 ipv6、优化系统内核、停掉一些没必要启动的系统服务等。此脚本尤其适合大批新安装的 Centos 系列的服务器。脚本代码如下所示（此脚本在 Centos5.5 x86_64 下已通过）：

```
#!/bin/bash
# Created by kerryhu
# MAIL:king_819@163.com
# BLOG:http://kerry.blog.51cto.com
cat << EOF
+ -----+
|           = = = Welcome to Centos System init = = =           |
+ -----+
+ -----by kerry-----+
EOF

#set ntp
yum -y install ntp
echo "* 3 * * * /usr/sbin/ntpdate 210.72.145.44 > /dev/null 2>&1" >> /etc/crontab
service crond restart
#set ulimit
echo "ulimit -SHn 102400" >> /etc/rc.local
#set locale
#true > /etc/sysconfig/i18n
#cat >> /etc/sysconfig/i18n<< EOF
#LANG="zh_CN.GB18030"
#SUPPORTED="zh_CN.GB18030:zh_CN:zh:en_US.UTF-8:en_US:en"
#SYSFONT="latarcyrheb-sun16"
#EOF
#set sysctl
true > /etc/sysctl.conf
cat >> /etc/sysctl.conf << EOF
net.ipv4.ip_forward = 0
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.default.accept_source_route = 0
kernel.sysrq = 0
kernel.core_uses_pid = 1
net.ipv4.tcp_syncookies = 1
kernel.msgmnb = 65536
kernel.msgmax = 65536
kernel.shmmax = 68719476736
kernel.shmall = 4294967296
net.ipv4.tcp_max_tw_buckets = 6000
net.ipv4.tcp_sack = 1
net.ipv4.tcp_window_scaling = 1
```



```

net.ipv4.tcp_rmem = 4096 87380 4194304
net.ipv4.tcp_wmem = 4096 16384 4194304
net.core.wmem_default = 8388608
net.core.rmem_default = 8388608
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
net.core.netdev_max_backlog = 262144
net.core.somaxconn = 262144
net.ipv4.tcp_max_orphans = 3276800
net.ipv4.tcp_max_syn_backlog = 262144
net.ipv4.tcp_timestamps = 0
net.ipv4.tcp_synack_retries = 1
net.ipv4.tcp_syn_retries = 1
net.ipv4.tcp_tw_recycle = 1
net.ipv4.tcp_tw_reuse = 1
net.ipv4.tcp_mem = 94500000 915000000 927000000
net.ipv4.tcp_fin_timeout = 1
net.ipv4.tcp_keepalive_time = 1200
net.ipv4.ip_local_port_range = 1024 65535
EOF
/sbin/sysctl -p
echo "sysctl set OK!!"
#close ctrl+alt+del
sed -i "s/ca::ctrlaltdel:\//sbin\//shutdown -t3 -r now/#ca::ctrlaltdel:\//sbin\//shutdown -t3
-r now/" /etc/inittab
#set purview
chmod 600 /etc/passwd
chmod 600 /etc/shadow
chmod 600 /etc/group
chmod 600 /etc/gshadow
#disable ipv6
cat << EOF
+ -----+
|           = = = Welcome to Disable IPV6 = = =           |
+ -----+
EOF
echo "alias net -pf -10 off" >> /etc/modprobe.conf
echo "alias ipv6 off" >> /etc/modprobe.conf
/sbin/chkconfig --level 35 ip6tables off
echo "ipv6 is disabled!"
#disable selinux
sed -i '/SELINUX/s/enforcing/disabled/' /etc/selinux/config
echo "selinux is disabled,you must reboot!"
#vim
sed -i "s/^/alias vi='vim'/" /root/.bashrc
echo 'syntax on' > /root/.vimrc
#zh_cn
sed -i -e 's/^LANG=.* /LANG="en"/' /etc/sysconfig/i18n
#init_ssh
ssh_cf="/etc/ssh/sshd_config"

```

```

sed -i -e '74 s/^/#/' -i -e '76 s/^/#/' $ssh_cf
#sed -i "s/#Port 22/Port 65535/" $ssh_cf
sed -i "s/#UseDNS yes/UseDNS no/" $ssh_cf
#client
sed -i -e '44 s/^/#/' -i -e '48 s/^/#/' $ssh_cf
service sshd restart
echo "ssh is init is ok....."
#chkser
#tunoff services
# -----
cat << EOF
+ ----- +
|           = = = Welcome to Tunoff services = = =           |
+ ----- +
EOF
# -----
for i in `ls /etc/rc3.d/S*`
do
    CURSRV='echo $i |cut -c 15 - '
echo $CURSRV
case $CURSRV in
    crond |irqbalance |microcode_ctl |network |random |sshd |syslog |local )
    echo "Base services, Skip!"
    ;;
    * )
    echo "change $CURSRV to off"
    chkconfig --level 235 $CURSRV off
    service $CURSRV stop
    ;;
esac
done
echo "service is init is ok....."

```

3. 自动同步局域网类 Centos yum 的镜像服务器脚本

尽管 Sohu 网站和 163 网站都推出了 mirrors 服务，但若要配置多台服务器时仍然会占用很多公网带宽，因此配置一个本地 yum 源并时常与公网同步就显得非常必要了。这里介绍的就是针对 Centos 各个版本配置 yum 源，并且每日同步公网数据。前提是必须先安装 rsync，推荐用最新版本。脚本内容如下（此脚本在 Centos5.5 x86_64 下已通过）：

```

#!/bin/bash
#author:coralzd powered by www.freebsdssystem.org
#description: yum rsync script
Date = `date +%Y%m%d`
LogFile="/var/log/rsync_yum/ $Date.log"
CentOSTrunkVer = "5"
CentOSCurrentVer = "5.5"
CentOSCurrentv = "4.8"
CentOSCurrentv2 = "5.6"
ReceiveMail = "admin@admin.com"

```

```

Cpanpath="/data0/yum/mirrors.sohu.com/CPAN/"
RsyncBin="/usr/local/bin/rsync" #rsync 路径,以自己安装的路径为准
RsyncPerm="-avrt --delete --exclude=debug/ --exclude=isos/"
CentOS_Trunk_Ver_Path="/data0/yum/mirrors.sohu.com/centos/ $CentOSTrunkVer"
CentOS_Current_Ver_Path="/data0/yum/mirrors.sohu.com/centos/ $CentOSCurrentVer"
CentOS_Current_v_Path="/data0/yum/mirrors.sohu.com/centos/ $CentOSCurrentv"
CentOS_Current_v_Path2="/data0/yum/mirrors.sohu.com/centos/ $CentOSCurrentv2"
YumSiteList="rsync://mirrors.sohu.com/centos"
Cpansitelist="cpan.wenzk.com::CPAN"
#===== epel =====
epelSite="rsync://mirrors.sohu.com/fedora-epel/"
epelLocalPath="/data0/yum/mirrors.sohu.com/epel"
#rpmforge
rpmforgeSite="rsync://mirrors.sohu.com/dag/redhat/"
rpmforgeLocalPath="/data0/yum/mirrors.dzwww.com/rpmforce"
#CPAN mirrors
cpansite="rsync://mirrors.sohu.com/nginx/"
cpanlocalpath="/data0/yum/mirrors.sohu.com/CPAN/"
echo "---- $Date 'date +%T' Begin ----" >> $LogFile # 输出开始时间到日志文件
#CPAN
$RsyncBin $RsyncPerm $Cpansitelist $cpanlocalpath >> $LogFile
# centos 5
$RsyncBin $RsyncPerm $YumSiteList/ $CentOSTrunkVer/
$CentOS_Trunk_Ver_Path >> $LogFile
# Centos 5.5
$RsyncBin $RsyncPerm $YumSiteList/ $CentOSCurrentVer/
$CentOS_Current_Ver_Path >> $LogFile
# Centos 4.8
$RsyncBin $RsyncPerm $YumSiteList/ $CentOSCurrentv/ \
$CentOS_Current_v_Path >> $LogFile
# Centos 5.6
$RsyncBin $RsyncPerm $YumSiteList/ $CentOSCurrentv2/ \
$CentOS_Current_v_Path2 >> $LogFile
# EPEL
$RsyncBin $RsyncPerm --exclude=4/ --exclude=4AS/ --exclude=4AS/ \
--exclude=4WS/ --exclude=beta/ --exclude=testing/ $epelSite $epelLocalPath
>> $LogFile
# RPMFORGE
$RsyncBin $RsyncPerm $rpmforgeSite $rpmforgeLocalPath >> $LogFile
# CPAN
#$RsyncBin $RsyncPerm $cpansite $cpanlocalpath >> $LogFile
echo "---- $Date 'date +%T' End ----" >> $LogFile #输出完成时间到日志文件

```

5.6.6 生产环境下的安全类 SHELL 脚本

我的几台公网机器放在电信的 IDC 机房内（没有硬件防火墙），被人不停地 SSH 连接，我查看了一下，其中有个 IP 居然连接失败 17 000 次了。后来环境部署成熟以后发现仍然有不少外网 IP 在扫描和试探，看来还是要做些安全防范措施。此时，线上环境已经可以很稳定地运行了，我不想安装 Denyhosts 防暴力破解工具。另外由于牵涉到了 LVS 服务器，如果运行 iptables 禁止 IP 的话，会

影响整个集群环境，所以我采取将非法 IP 写入/etc/hosts.deny 文件的方法。

这里补充一下关于/etc/hosts.deny 的小知识：在 Linux 系统中，/etc/hosts.allow 控制可以访问本机的 IP 地址，/etc/hosts.deny 控制禁止访问本机的 IP。如果两个文件的配置有冲突，以/etc/hosts.deny 为准。如果只有/etc/hosts.deny 文件的话，那么此文件的 IP 会全部被禁止访问。

脚本的最初思路是每隔一段时间就收集一次/var/log/secure 的日志信息。后来觉得这个方法比较复杂。我稍微解释一下下面的脚本：它其实就放在 Crontab 里，每隔一小时去读一下/var/log/secure 的日志信息，定义连接 root 的 IP 阈值为 10，如果此 IP 在/etc/hosts.deny 里，就什么也不做；如果不在，就将其添加至/etc/hosts.deny 里。脚本代码如下（此脚本在 Centos5.5 x86_64 下已通过，我目前所有机房的线上服务器都配置了此安全脚本）：

```
#!/bin/bash
#Denyhosts SHELL SCRIPT
#2011-04-01

cat /var/log/secure |awk '/Failed/{print $(NF-3)}'|sort |uniq -c |awk

'{print $2"=" $1;}' >/root/black.txt
DEFINE="10"
for i in `cat /root/black.txt`
do
    IP=`echo $i |awk -F= '{print $1}'`
    NUM=`echo $i |awk -F= '{print $2}'`
    if [ $NUM -gt $DEFINE ];
    then
        grep $IP /etc/hosts.deny > /dev/null
        if [ $? -gt 0 ];
        then
            echo "sshd: $IP" >> /etc/hosts.deny
        fi
    fi
done
```

这里解释下 \$? 命令。当一个脚本退出时，\$? 命令保存脚本的退出状态码，这个退出状态码也就是脚本中最后一个执行命令的退出状态码。一般情况下，0 表示成功，1~255 的整数表示错误。将此代码放进/etc/crontab 里，每分钟执行一次。我初期本来是想用截取时间段的方法来做，但整个取样过程极为繁琐，反而不如用 Crontab 收集即时/var/log/secure 日志来得方便，如下所示：

```
* /1 * * * root sh /root/ssh_deny.sh
```

运行一段时间后，会将某台公网服务器上某一分钟内 SSH 过多次的 IP（比如超过 10 次）写进/etc/hosts.deny 里进行封锁。当然了，大家可以根据自己线上服务器的情况更改此阈值。black.txt 为单位时间内 SSH 连接次数过多的 IP，我们可以用 cat 命令查看一下，如下所示：

```
cat black.txt
114.34.180.134=26
122.154.29.123=37
125.88.128.158=31
174.143.209.190=35
```



```

202.71.107.146=36
203.217.188.105=49
203.67.31.148=83
206.16.140.23=7
221.132.34.145=26
60.166.9.104=27
61.230.64.149=30
83.149.32.228=26
93.187.187.111=23
93.62.118.35=24

```

超过 10 次的 IP，程序会自动将其添加至 `/etc/hosts.deny` 中，达到封锁的目的。我们可以等脚本运行一段时间后，查看一下 `/etc/hosts.deny` 文件的内容，命令如下：

```

cat /etc/hosts.deny
sshd:121.9.205.79
sshd:88.190.16.9
sshd:202.39.33.34
sshd:91.148.134.59
sshd:202.43.151.3
sshd:173.193.216.103
sshd:112.125.51.191
sshd:218.75.21.26
sshd:202.111.152.12
sshd:77.221.156.210
sshd:189.27.4.115
sshd:189.204.49.48
sshd:114.34.180.134
sshd:203.217.188.105
sshd:93.187.187.111
sshd:203.67.31.148
sshd:61.230.64.149
sshd:83.149.32.228
sshd:174.143.209.190
sshd:221.132.34.145
sshd:122.154.29.123
sshd:93.62.118.35
sshd:60.166.9.104

```

以上证明脚本生效了，并且在认真负责地工作，这样我们的服务器是不是更轻松点？互联网上的恶意 SSH 暴力破解还是非常多的，如果我们的 Linux 服务器密码过于简单，而我们的系统管理员又忘了定期更改 root 密码的话，那么迟早有一天我们的 root 密码会被暴力破解。用了以上的脚本后，我们再也不用担心这个问题了。

另外，我朋友在工作中遇到了一个问题，大家有可能也会遇到，这里把相关情况说明一下：朋友的业务服务器是放在某电信 IDC 机房内托管的，网站相关的服务器都连在同一台交换机上，网关为 203.93.236.129。有一天迁进来两台 Windows Server 2003 的机器后（也连进了同一台交换机），却发现客户访问朋友网站的速度很慢，丢包频繁。

朋友立即打电话给我，我怀疑是 Windows Server 2003 感染了病毒从而攻击网关，于是尝试在同

一交换机的某台 Centos5.5 机器下 ping 203.93.236.129，很惊奇地发现了 udp！报错。这说明我的网段中存在着另一个重复地址 203.93.236.129，也就是说网关被人攻击了。我们该如何找出这台机器呢？其实可以用如下脚本 arp.sh 来找，代码如下：

```
#!/bin/bash
for i in {120..150}
do
arping -I eth0 203.93.236.$i -c 1
done
arp -a > mac.txt
```

我很久之前就做了 arp 的对应关系，正常网关 203.93.236.129 的 mac 地址我们可以用如下命令查找：

```
arp
```

arp 命令的显示结果如下：

Address	HWtype	HWaddress	Flags	Mask	Iface
203.93.236.129	ether	00:18:74:14:BE:80	C		eth0

我们可对比一下 mac.txt，它清楚地记录了每台 IP 和 mac 地址的对应关系，我们应该会发现那台感染病毒的机器，并将其迅速隔离，免得影响整个网络。上面的脚本其实就是通过一个循环，使用 arping 来对所有的机器（即 IP 在 203.93.236.120-203.93.236.150 下面的机器）发一个包，这样就可以在 arp 下面查看到相应的 mac 缓存，进而得到对应的 IP 地址了。我们可以关注跟网关 IP 相同的机器，它肯定是有问题的机器。后来，通过这种方法我们找出一台 203.93.236.149 的 Windows Server 2003 服务器感染了病毒，所以立即将其下线并将其隔离，避免影响了整个网络机器的业务。

另外，我们在工作中还会将防火墙的安全策略写成 SHELL 脚本，这样无论是调试还是维护都是件非常方便和容易的事情。由于这些代码主要还是 iptables 的语法，所以我在后面的 Linux 防火墙相关章节里进行了详细说明，这里就不举例说明了。

5.7 小结

本章向大家详细说明了 Vim 的使用语法，以及 Sed 在日常工作中的使用案例，并用 SHELL 命令 grep 和 find 结合正则表达式说明了正则表达式的一些基础用法。在后面的 SHELL 脚本实例中，我从备份、监控、统计、自动化、开发、安全几方面向大家演示了在生产环境下我们经常用到的 SHELL 脚本。我们在感叹 SHELL 脚本强大的管理功能的同时，也应该比较清楚 SHELL 脚本在开发功能上的不足。所以我在这里向大家推荐 Python，它继承了传统编译语言的强大性和通用性，同时也借鉴了简单脚本和解释语言的易行性，运行速度也不慢，适合网站开发，正好可以弥补 SHELL 脚本的不足。结合这两种脚本语言，我们的系统管理和代码开发工作会更加得心应手。



第 6 章 构建高可用的Linux集群

- 6.1 负载均衡高可用的核心概念和常用软件
- 6.2 负载均衡中的名词解释
- 6.3 负载均衡器的会话保持机制
- 6.4 Linux 集群的项目案例分享
- 6.5 项目实践中 Linux 集群的总结和思考
- 6.6 网站架构应关注和研究的方向
- 6.7 MySQL 数据库的优化
- 6.8 生产环境下的 MySQL 数据库备份
- 6.9 部分项目施工图纸
- 6.10 小结

作为一名 Linux/Unix 系统工程师和项目实施工程师，我在工作中经常会遇到一些对外项目，比如小中型网站的架构及实施。在为客户实施项目方案时，客户基本上都会提出这样一个要求：用 Linux 集群，保证服务的高可用性。基于此，我们所有的服务器，包括负载均衡器、文件服务器、Web 服务器和 MySQL 数据库，基本上都有两台或两台以上的服务器。而且根据客户的要求及客户自身机房的硬件配置，我们还会选择不同的负载均衡器，比如硬件有 F5，软件有 LVS、Nginx 及 HAProxy。在相当长的一段时间内，我的工作就是不停地测试它们，不停地完善和优化整体网站的架构。

我在与一些系统管理员线下交流时发现，不少技术优秀的系统管理员由于公司环境等因素的制约，他们对负载均衡、Linux 集群等相关知识知之甚少。在这里，我想与大家分享下自己在 Linux 集群方面的一些项目经验，希望能帮助大家理解和掌握 Linux 集群及其高可用方面的知识。

6.1 负载均衡高可用的核心概念和常用软件

6.1.1 什么是负载均衡高可用

在解释“负载均衡高可用”这个专业术语之前，我们先搞清楚一个问题：我们为什么需要负载均衡？我们先看一个例子，假如我们有一个资讯类的网站，只允许 100 个用户同时在线访问。网站上线初期，由于知名度较小，通常只有几个用户经常上线；后期知名度提高了，百度和谷歌又收录了我们的网站，于是同时在线的用户数量直线上升，很快达到上千人。于是，网站负荷加重，经常会“反应迟钝”，这时用户开始埋怨。为了不影响用户对我们的信心，就一定要想办法解决这个问题。

试想，如果有几台或几十台相同配置的机器，前端放一个转发器，轮流转发客户对网站的请求，每台机器都将用户数控制在 100 之内，那么网站的反应速度就会大大提高，即使其中的某台服务器因为硬件故障宕机了，也不会影响用户的访问。其中，这个神奇的转发器就是负载均衡器 (Director)。

那什么是负载均衡呢？负载均衡建立在现有的网络结构之上，它提供了一种廉价、有效、透明的方法来扩大网络设备和服务器的带宽、增加吞吐量、加强网络数据处理能力，以及提高网络的灵活性和可用性。通过负载均衡器，我们可以实现 N 台廉价的 Linux 服务器并行处理，从而达到小型机或大型机的计算能力，这也是为何负载均衡如此流行的主要原因。

“高可用” (High Availability, HA) 其实有两种不同的含义：广义来讲，是指整个系统的高可用性；狭义来讲，一般指主机的冗余接管，如主机 HA。如无特殊说明，本书中的 HA 都是指广义的高可用性。广义的高可用性可保证整个系统不会因为某一台主机崩溃或发生故障而出现停止服务的现象，狭义的高可用性则要从最前端的负载均衡器谈起了。

单台负载均衡器位于网站的最前端，它起着分流客户请求的作用，相当于整个网站或系统的入口，如果它出现故障，整个网站也会出现故障。所以，这时我们要有一种方案，它能在短时间（一般要求小于 1 秒）内将崩溃的负载均衡器接管过去，这也称为高可用。至于负载均衡器后端的 Web 集群、数据库集群，因为有负载均衡器的内部机制，即使其中的某一台或两台发生问题，也不会影响整套系统的使用。

我们现在俗称的 Linux 集群，它指的是大范围内的整套系统架构，相对于负载均衡器后端的 Web 集群、Tomcat 集群或 MySQL 集群来说，它的涵盖面要广得多，包括了负载均衡高可用。为了

便于区别，我在提到集群时一般会带上前缀，比方说 Web 集群，所指的是后端提供相同服务的 Web 机器群；如果是 Linux 集群，则指的是大范围的系统集群架构，希望大家不要混淆。

目前，在线上环境中应用得较多的负载均衡器硬件有 F5 BIG-IP，软件有 LVS、Nginx 及 HA-Proxy，高可用软件有 Heartbeat、Keepalived，成熟的 Linux 集群架构有 LVS + Keepalived、Nginx + Keepalived 及 DRBD + Heartbeat。

6.1.2 以 F5 BIG-IP 作为负载均衡器

作为负载均衡器的硬件主要有 F5 BIG-IP 和 Citrix NetScaler，CDN 机房中最常见的硬件负载均衡设备就是大名鼎鼎的 F5 BIG-IP。

我在项目实施过程中主要使用的也是 F5 BIG-IP，这里简单介绍一下它。F5 BIG-IP LTM 的官方名称叫做本地流量管理器，可以做 4~7 层的负载均衡，具有负载均衡、应用交换、会话交换、状态监控、智能网络地址转换、通用持续性、响应错误处理、IPv6 网关、高级路由、智能端口镜像、SSL 加速、智能 HTTP 压缩、TCP 优化、第 7 层速率整形、内容缓冲、内容转换、连接加速、高速缓存、Cookie 加密、选择性内容加密、应用攻击过滤、拒绝服务（DoS）攻击和 SYN Flood 保护、包过滤防火墙、包消毒等功能。

以下是 F5 BIG-IP 用作 HTTP 负载均衡器时的主要功能：

(1) 提供了 12 种灵活的算法将所有流量均衡地分配到各个服务器，对用户而言，它只是一台虚拟服务器。

(2) 可以确认应用程序能否针对请求返回相应的数据。假如 F5 BIG-IP 后面的某一台服务器发生服务停止和死机等故障，它会检查出来并将该服务器标识为宕机，从而不会将用户的访问请求传送到这台发生故障的服务器上。只要其他的服务器正常，用户的访问就不会受到影响。宕机的服务器一旦修复，F5 BIG-IP 就会自动查证，当确认其能对客户请求作出正确响应时即恢复向该服务器传送请求。

(3) 具有动态 Session 的会话保持功能。

6.1.3 以 LVS 作为负载均衡器

说明：以下资料源自 LVS 官方网站。

LVS 全称为 Linux Virtual Server，它是章文嵩博士（现淘宝网基础核心软件研发负责人）主持的自由软件项目。LVS 是一个负载均衡/高可用性集群，主要针对大业务量的网络应用（如新闻服务、网上银行、电子商务等）。它建立在一个主控服务器（通常为双机）及若干真实服务器（Real-server）所组成的集群之上。Real-server 负责实际提供服务，主控服务器根据指定的调度算法对 Real-server 进行控制。而集群的结构对于用户来说是透明的，客户端只与单个的 IP（集群系统的虚拟 IP）进行通信，也就是说从客户端的视角来看，这里只存在单个服务器。

Real-server 可以提供众多服务，如 FTP、HTTP、DNS、Telnet、SMTP，另外，现在比较流行将其用于 MySQL 集群。主控服务器负责对 Real-server 进行控制。客户端在向 LVS 发出服务请求时，Director 会通过特定的调度算法来指定由某个 Real-server 去应答请求，而客户端只与 Load Balancer 的 IP（即虚拟的 VIP）进行通信。如果以上工作流程用 LVS 的工作拓扑来说明，效果可能更好，如图 6-1 所示。

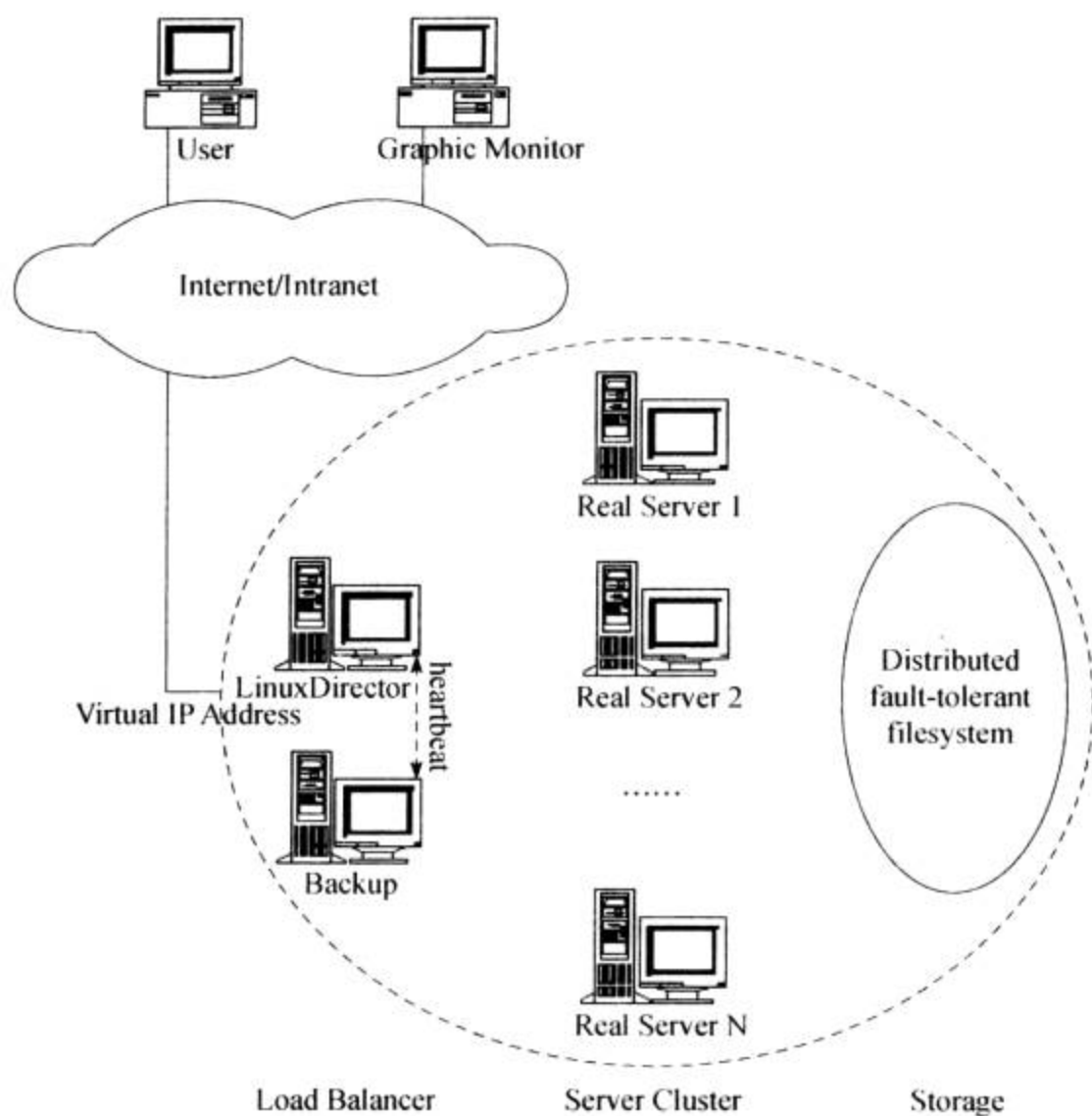


图 6-1 LVS 集群的体系结构

1. LVS 集群的体系结构

在设计 LVS 集群时需要考虑系统的透明性、可伸缩性、高可用性和易管理性。一般情况下，LVS 集群采用 3 层结构，主要组成部分如下所示。

- 负载调度器（load balancer）：它是整个集群对外的前端机，负责将用户的请求发送到一组服务器上执行，而用户认为服务是来自一个 IP 地址（我们可称之为虚拟 IP 地址）上的。
- 服务器池（server pool）：它是一组真正执行客户请求的服务器，执行的服务有 Web、Mail、FTP 和 DNS 等。
- 共享存储（shared storage）：它为服务器池提供一个共享的存储区，这样很容易使得服务器池拥有相同的内容，提供相同的服务。

调度器是服务器集群系统的唯一入口点（Single Entry Point），它可以采用 IP 负载均衡技术和基于内容的请求分发技术，或将两者相结合。在 IP 负载均衡技术中，要求服务器池拥有相同的内容，提供相同的服务。当用户请求到达时，调度器只根据服务器的负载情况和设定的调度算法从服务器池中选出一个服务器，将该请求转发到选出的服务器上，并记录这一次调度。当这个请求的其他报文到达时，也会被转发到前面选出的服务器上。在基于内容的请求分发技术中，服务器可以提供不同的服务，当用户请求到达时，调度器可根据请求的内容选择执行请求的服务器。因为所有的操作都是在 Linux 操作系统的核心空间中完成的，它的调度开销很小，所以它具有很高的吞吐率。

服务器池的节点数目是可变的。当整个系统的负载超过目前所有节点的处理能力时，可以在服务器池中增加服务器来满足不断增长的请求负载。对大多数网络服务来说，请求间的相关性并不是很强，请求可以在不同的节点上并行执行，所以整个系统的性能基本上可以随着服务器池的节点数目的增加而线性增长。

共享存储通常是数据库、网络文件系统或分布式文件系统。服务器节点需要动态更新的数据一般存储在数据库系统中，同时数据库会保证并发访问时数据的一致性。静态的数据可以存储在网络文件系统（如 NFS/CIFS）中，但网络文件系统的伸缩能力有限，一般来说，NFS/CIFS 服务器只能支持 3~6 个繁忙的服务器节点。规模较大的集群系统可以考虑用分布式文件系统，如 AFS、GFS、Coda 和 Intermezzo 等。分布式文件系统可以为各服务器提供共享的存储区，它们访问分布式文件系统就像访问本地文件系统一样。同时，分布式文件系统可提供良好的伸缩性和可用性。

此外，当不同服务器上的应用程序同时访问分布式文件系统上的同一资源时，只有解决应用程序的访问冲突才能使得资源处于一致状态。此时需要一个分布式锁管理器（Distributed Lock Manager），它可能是分布式文件系统内部提供的，也可能是外部提供的。开发者在写应用程序时，可以使用分布式锁管理器来保证应用程序在不同节点上并发访问的一致性。

负载调度器、服务器池和共享存储系统之间通过高速网络相连接，如 100Mbps 交换网络、Myrinet 和 Gigabit 网络等。之所以要求使用高速网络，主要是为了避免当系统规模扩大时网速成为整个系统的瓶颈。

Graphic Monitor 是为系统管理员提供整个集群系统状态的监视器，它可以监视系统的状态。因为 Graphic Monitor 是基于浏览器的，所以管理员可以随时随地监测系统的状况。考虑到安全原因，浏览器要通过 HTTPS（Secure HTTP）协议和身份认证后才能进行系统监测，并进行系统的配置和管理。

2. 通过 NAT 实现虚拟服务器（VS/NAT）

由于 IPv4 中 IP 地址的日益紧张以及出于安全方面的考虑，很多网络使用保留 IP 地址（如 10.0.0.0/255.0.0.0、172.16.0.0/255.128.0.0 和 192.168.0.0/255.255.0.0）。这些地址不在 Internet 上使用，而是专门为内部网络预留的。当内部网络中的主机要访问 Internet 或被 Internet 访问时，就需要进行网络地址转换（Network Address Translation, NAT），将内部地址转换为 Internet 上可用的外部地址。NAT 的工作原理是报文头（目标地址、源地址和端口等）被正确改写后，客户端相信它们连接的是同一个 IP 地址，而不同 IP 地址的服务器组也认为它们是与客户直接相连的。由此，可以用 NAT 方法将不同 IP 地址的并行网络服务变成一个在同一 IP 地址上的虚拟服务。

VS/NAT 的体系结构比较简单：在一组服务器前有一个调度器，它们是通过 Switch/HUB 相连接的。这些服务器提供相同的网络服务、相同的内容，即不管请求被发送到哪一台服务器，执行结果都是一样的。服务的内容可以复制到每台服务器的本地硬盘上，可以通过网络文件系统（如 NFS）共享，也可以通过一个分布式文件系统来提供。

用户通过 Virtual IP Address（虚拟服务的 IP 地址）访问网络服务时，请求报文到达调度器，调度器根据连接调度算法从一组真实服务器中选出一台服务器，将报文的目标地址 Virtual IP Address 改写成选定服务器的地址，将报文的目标端口改写成选定服务器的相应端口，最后将修改后的报文发送给选出的服务器。同时，调度器在连接 Hash 表中记录这个连接，当这个连接的下一个报文到达时，从连接 Hash 表中可以得到原先选定的服务器的地址和端口，执行同样的改写操作，并将报

文传给原先选定的服务器。当来自真实服务器的响应报文经过调度器时，调度器将报文的源地址和源端口改为 Virtual IP Address 和相应的端口，再把报文发给用户。

我们在连接上引入一个状态机，不同的报文会使得连接处于不同的状态，不同的状态有不同的超时值。在 TCP 连接中，根据标准的 TCP 有限状态机进行状态迁移，这里我们不详细叙述，请参见 W. Richard Stevens 所著的《TCP/IP Illustrated Volume I》。在 UDP 中，我们只设置一个 UDP 状态。不同状态的超时值是可以设置的，在默认情况下，SYN 状态的超时时间为 1 分钟，ESTABLISHED 状态的超时时间为 15 分钟，FIN 状态的超时时间为 1 分钟，UDP 状态的超时时间为 5 分钟。当连接终止或超时，调度器会将这个连接从连接 Hash 表中删除。

这样，用户所看到的只是在 Virtual IP Address 上提供的服务，而服务器集群的结构对用户而言是透明的。针对改写后的报文，应用增量调整 Checksum 的算法调整 TCP Checksum 的值，避免了扫描整个报文来计算 Checksum 的开销。

3. 通过 IP 隧道实现虚拟服务器 (VS/TUN)

在 VS/NAT 的集群系统中，请求和响应的数据报文都需要通过负载调度器，当真实服务器的数量在 10 台和 20 台之间时，负载调度器将成为整个集群系统的新瓶颈。大多数 Internet 服务都有这样的特点：请求报文较短而响应报文往往包含大量的数据。如果能将请求和响应分开处理，即负载调度器只负责调度请求，而响应直接返回给客户，这将极大地提高整个集群系统的吞吐量。

IP 隧道 (IP tunneling) 是将一个 IP 报文封装到另一个 IP 报文中的技术，这可以使得目标为一个 IP 地址的数据报文能被封装和转发到另一个 IP 地址。IP 隧道技术也称为 IP 封装技术 (IP encapsulation)，主要应用于移动主机和虚拟私有网络 (Virtual Private Network)，在其中隧道都是静态建立的，隧道一端有一个 IP 地址，另一端也有一个唯一的 IP 地址。

我们利用 IP 隧道技术将请求报文封装并转发给后端服务器，响应报文能从后端服务器直接返回给客户。但是此时后端服务器是一组而非一个，所以我们不可能静态地建立一一对应的隧道，而是动态地选择一台服务器，将请求报文封装并转发给选出的服务器。这样，我们可以利用 IP 隧道的原理将一组服务器上的网络服务组成在一个 IP 地址上的虚拟网络服务。

4. 通过直接路由实现虚拟服务器 (VS/DR)

与 VS/TUN 方法相同，VS/DR 利用了大多数 Internet 服务的非对称特点，负载调度器只负责调度请求，而服务器直接将响应返回给客户，可以极大地提高整个集群系统的吞吐量。该方法与 IBM 的 NetDispatcher 产品中使用的方法类似（其中服务器上的 IP 地址的配置方法是相似的），但 IBM 的 NetDispatcher 是非常昂贵的商业化产品，我们无法知道它的内部机制，其中有一些是 IBM 的专利。

VS/DR 的体系结构比较简单：调度器和服务器组都必须在物理上有一个网卡通过局域网相连，如通过高速的交换机或者 HUB 相连。VIP 地址被调度器和服务器组共享，调度器配置的 VIP 地址是对外可见的，用于接收虚拟服务的请求报文。所有的服务器把 VIP 地址配置在各自的 Non-ARP 网络设备上，它对外是不可见的，只是用于处理目标地址为 VIP 的网络请求。

VS/DR 的连接调度和管理与 VS/NAT 和 VS/TUN 一样，但它的报文转发方法不同，它将报文直接路由给目标服务器。在 VS/DR 中，调度器根据各个服务器的负载情况，动态地选择一台服务器，不修改也不封装 IP 报文，而是将数据帧的 MAC 地址改为选出服务器的 MAC 地址，再将修改后的数据帧在服务器组所在的局域网内发送（这也是 LVS/DR 要求服务器在同一个局域网内的原因）。因为数据帧的 MAC 地址是选出的服务器，所以服务器肯定可以收到这个数据帧，从中可以获得该

IP 报文。当服务器发现报文的目标地址（VIP 地址）是在本地的网络设备上，服务器处理这个报文后，就会根据路由表将响应报文直接返回给客户。

在 VS/DR 中，根据默认的 TCP/IP 协议栈处理，请求报文的目标地址为 VIP，响应报文的源地址肯定也为 VIP，所以响应报文不需要作任何修改，可以直接返回给客户，客户得到正常的服务，但不会知道是哪一台服务器处理的。

VS/DR 负载调度器与 VS/TUN 一样，只处于从客户到服务器的半连接中，按照半连接的 TCP 有限状态机进行状态迁移。

5. 3 种方法的优缺点比较

(1) VS/NAT 的优缺点

VS/NAT 的优点是服务器可以运行任何支持 TCP/IP 的操作系统，它只需要一个 IP 地址配置在调度器上，服务器组可以用私有的 IP 地址。缺点是它的伸缩能力有限，当服务器节点数目增加到 20 时，调度器本身有可能成为系统的新瓶颈，因为在 VS/NAT 中请求和响应报文都需要通过负载调度器。很早以前，我们在 Pentium 166 处理器的主机上测得重写报文的平均延时为 60us，性能更高的处理器上延时会短一些。假设 TCP 报文的平均长度为 536Bytes，则调度器的最大吞吐量为 8.93MBytes/s。我们再假设每台服务器的吞吐量为 800KBytes/s，这样一个调度器可以带动 10 台服务器。

基于 VS/NAT 的集群系统可以满足很多服务器的性能要求。如果负载调度器成为系统的新瓶颈，有 3 种解决方法：混合方法、VS/TUN 和 VS/DR。在 DNS 混合集群系统中，有若干个 VS/NAT 负载调度器，每个负载调度器带有自己的服务器集群，同时这些负载调度器又通过 RR-DNS 组成简单的域名。但是，VS/TUN 和 VS/DR 是提高系统吞吐量的更好方法。

对于那些将 IP 地址或端口号放在报文数据中传送的网络服务，需要编写相应的应用模块来转换报文数据中的 IP 地址或端口号。这会带来较大的工作量，同时应用模块检查报文的开销会降低系统的吞吐率。

(2) VS/TUN 的优点

在 VS/TUN 的集群系统中，负载调度器只将请求调度到不同的后端服务器，后端服务器将应答的数据直接返回给用户。这样，负载调度器就可以处理大量的请求，它甚至可以调度百台以上的服务器（同等规模的服务器），但它不会成为系统的瓶颈。即使负载调度器只有 100Mbps 的全双工网卡，整个系统的最大吞吐量也可超过 1Gbps。所以，VS/TUN 可以极大地增加负载调度器调度的服务器数量。VS/TUN 调度器可以用来构建高性能的超级服务器。

VS/TUN 技术对服务器有要求，即所有的服务器必须支持 IP Tunneling 或 IP Encapsulation 协议。目前，VS/TUN 的后端服务器主要运行 Linux 操作系统，我们暂时没有对其他操作系统进行测试。因为 IP Tunneling 正成为各个操作系统的标准协议，所以 VS/TUN 应该会适用于运行其他操作系统的后端服务器。

(3) VS/DR 的优点

与 VS/TUN 方法一样，VS/DR 调度器只处理客户到服务器端的连接，响应数据可以直接从独立的网络路由返回给客户，这可以极大地提高 LVS 集群系统的可伸缩性。

与 VS/TUN 相比，这种方法没有 IP 隧道的开销，但是要求负载调度器与实际服务器都有一块网卡连在同一物理网段上，服务器网络设备（或者设备别名）不作 ARP 响应，或者能将报文重定

向 (Redirect) 到本地的 Socket 端口上。

由于 VS/DR 的转发效率是最高的, 因此我们的项目多采用的是 VS/DR 模式, 而且网站服务器置于同一机房, 也方便维护和调试, 希望大家首先掌握此种用法。

6. LVS 算法介绍

在转发方式选定的情况下, 采用哪种调度算法将决定整个负载均衡的性能表现。不同的算法适用于不同的生产环境, 有时可能需要针对特殊需求自行设计调度算法。LVS 的算法是逐渐丰富起来的, 最初 LVS 只提供 4 种调度算法, 现在已经发展到 12 种, 我们下面来看其中重要的 8 种。

(1) 轮叫调度 (Round Robin)

调度器通过“轮叫”调度算法将外部请求按顺序轮流分配到集群中的真实服务器上, 它均等地对待每一台服务器, 而不管服务器上实际的连接数和系统负载。任何形式的负载均衡器 (包括硬件或软件级别的) 都带有基本的轮叫 (也叫轮询功能)。

(2) 加权轮叫 (Weighted Round Robin)

调度器通过“加权轮叫”调度算法根据真实服务器的不同处理能力来调度访问请求。这样可以保证处理能力强的服务器能处理更多的访问请求。调度器可以自动问询真实服务器的负载情况, 并动态地调整其权值。

(3) 最少连接 (Least Connections)

调度器通过“最少连接”调度算法动态地将网络请求调度到已建立的连接数最少的服务器上。如果集群系统的真实服务器具有相近的系统性能, 采用“最小连接”调度算法可以较好地均衡负载。

(4) 加权最少连接 (Weighted Least Connections)

在集群系统中的服务器性能差异较大的情况下, 调度器采用“加权最少连接”调度算法优化负载均衡性能, 具有较高权值的服务器将承受较大比例的活动连接负载。调度器可以自动问询真实服务器的负载情况, 并动态地调整其权值。

(5) 基于局部性的最少连接 (Locality-Based Least Connections)

“基于局部性的最少连接”调度算法是针对目标 IP 地址的负载均衡, 目前主要用于 Cache 集群系统。该算法根据请求的目标 IP 地址找出该目标 IP 地址最近使用的服务器, 若该服务器是可用的且没有超载, 则将请求发送到该服务器; 若服务器不存在, 或者该服务器超载且有服务器处于一半的工作负载, 则用“最少连接”的原则选出一个可用的服务器, 将请求发送到该服务器。

(6) 带复制的基于局部性的最少连接 (Locality-Based Least Connections with Replication)

“带复制的基于局部性的最少连接”调度算法也是针对目标 IP 地址的负载均衡, 目前主要用于 Cache 集群系统。它与 LBLC 算法的不同之处是它要维护的是从一个目标 IP 地址到一组服务器的映射, 而 LBLC 算法维护的是从一个目标 IP 地址到一台服务器的映射。该算法根据请求的目标 IP 地址找出该目标 IP 地址对应的服务器组, 按“最小连接”原则从服务器组中选出一台服务器, 若服务器没有超载, 将请求发送到该服务器; 若服务器超载, 则按“最小连接”原则从这个集群中选出一台服务器, 将该服务器加入到服务器组中, 将请求发送到该服务器。同时, 当该服务器组有一段时间没有被修改, 将最忙的服务器从服务器组中删除, 以降低复制的程度。

(7) 目标地址散列 (Destination Hashing)

“目标地址散列”调度算法根据请求的目标 IP 地址, 作为散列键 (Hash Key) 从静态分配的散

列表找出对应的服务器，若该服务器是可用的且未超载，将请求发送到该服务器，否则返回空。

(8) 源地址散列 (Source Hashing)

“源地址散列”调度算法根据请求的源 IP 地址，作为散列键 (Hash Key) 从静态分配的散列表找出对应的服务器，若该服务器是可用的且未超载，将请求发送到该服务器，否则返回空。

了解这些算法的原理有助于我们在特定的应用场合选择最适合的调度算法，从而尽可能地保持真实服务器的最佳性能。

6.1.4 以 Nginx 作为负载均衡器

Nginx 既可作为负载均衡器也可作为反向代理服务器，其配置方法相当简单，可以按轮询、IP_hash、URL_hash、权重等多种方法对后端的服务器执行负载均衡操作，同时还支持对后端服务器的健康检查。另外，相对于 LVS 来说，它有一个优势：由于它是基于第 7 层的负载均衡，是根据报头内的信息来执行负载均衡任务的，所以对网络的依赖性比较小，理论上只要 ping 得通就能够实现负载均衡。Nginx 不仅可以作为一款性能优异的负载均衡器，同时也是一款适用于高并发环境的 Web 应用软件，新浪、金山、迅雷等大型网站都有相关应用。其作为负载均衡器的优点如下：

(1) 配置文件非常简单，通俗易懂。

(2) 成本低廉。Nginx 是开源软件，可以免费使用，购买 F5 BIG-IP 和 NetScaler 等硬件负载均衡交换机则需要十几万甚至数十万人民币。

(3) 支持 Rewrite 重写规则。能够根据域名和 URL 的不同将 HTTP 请求分发到不同的后端服务器群组上。

(4) 有内置的健康检查功能。如果 Nginx Proxy 后端的某台 Web 服务器宕机了，不会影响前端访问。

(5) 节省带宽。支持 GZIP 压缩，可以添加浏览器本地缓存的 Header 头。

(6) 稳定性高。用于反向代理，宕机的概率极小。通过跟踪一些使用 Nginx 作为负载均衡器/反向代理服务器的已上线的项目，我们发现在高并发的情况下服务器宕机的次数几乎为零。

Nginx 的缺点是目前只支持 HTTP 和 Mail 的负载均衡。不过我们可以取长补短，根据其支持 Rewrite 重写规则和稳定性高的特点，可将其应用于大型网站的中间级别的负载均衡层，如图 6-2 所示。

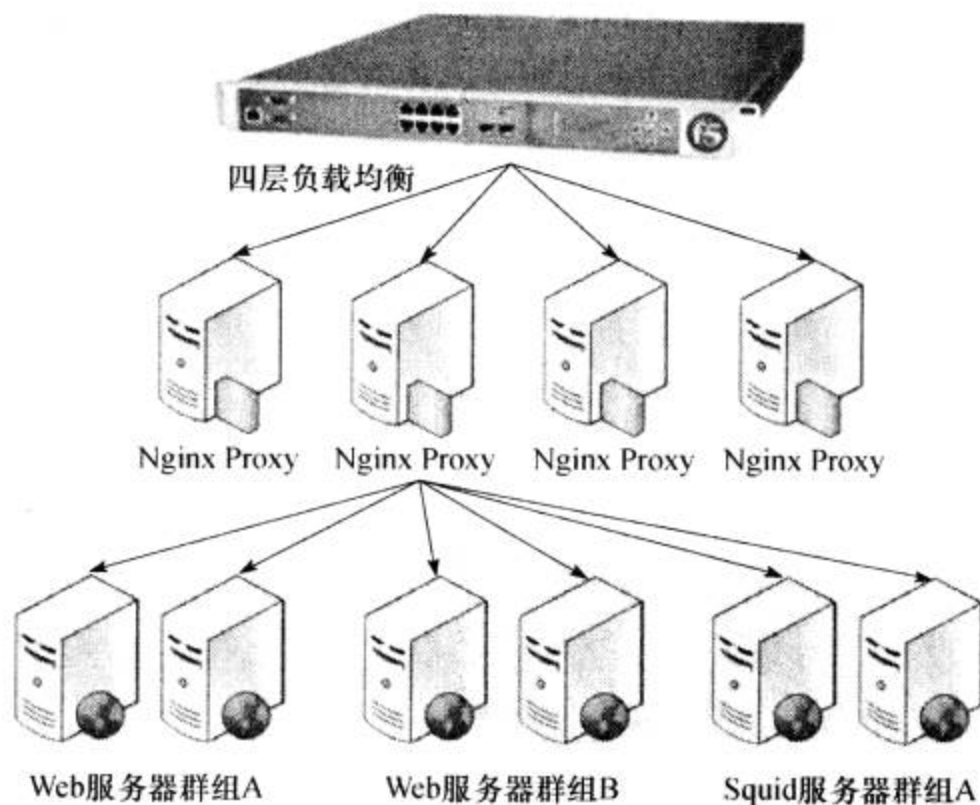


图 6-2 Nginx 作为中层负载均衡器的拓扑图

6.1.5 以 HAProxy 作为负载均衡器

HAProxy 是一款基于 TCP（第四层）和 HTTP（第七层）应用的代理软件，它也可作为负载均衡器，而且完全免费。借助 HAProxy，可以快速并且可靠地提供基于 TCP 和 HTTP 应用的代理解决

方案。HAProxy 最主要的优点是性能突出，它特别适合那些负载特大的 Web 站点，这些站点通常需要具备会话保持或七层处理功能。HAProxy 完全可以支持数以万计的并发连接，而且它的运行模式可以让你简单而安全地将它整合到你当前的架构中，同时可以保护你的 Web 服务器不暴露到网络上（通过防火墙 80 端口映射的方法）。作为一款优秀的负载均衡软件，HAProxy 优点如下所示：

(1) 免费且开源，稳定性也非常好，我在自己所做的一些小项目中发现，单 HAProxy 也运行得不错，其稳定性可以与硬件级的 F5 Big-IP 相媲美。

(2) 根据官方文档可知，HAProxy 可以跑满 10Gbps，对于软件级负载均衡器而言，这个数字是相当惊人的。

(3) 支持连接拒绝。因为维护一个连接保持打开状态的开销是很低的，有时我们需要防止蠕虫攻击，也就是通过限制它们的连接打开来防止它们的危害。这个功能已经拯救了很多被 DDoS 攻击的小型站点，这也是其他负载均衡器所不具备的。

(4) 支持全透明代理（已具备硬件防火墙的典型特点）。可以用客户端 IP 地址或任何其他地址来连接后端服务器，这个特性仅在 Linux 2.4/2.6 内核打了 cttproxy 补丁后才可以使使用。这个特性使得为某特殊服务器处理部分流量的同时又不修改服务器的地址成为可能。

(5) HAProxy 现在多用于线上的 MySQL 集群环境，常用它作为 MySQL（读）负载均衡。当然，如果将其作为最前端的 Web 负载均衡器也是相当稳定的。

(6) 自带强大的监控服务器状态的页面，在实际环境中我们可以结合 Nagios 来实现邮件或短信报警，这也是很多人非常喜欢它的原因之一。

(7) HAProxy 对 https 支持得相当好，证书文件可以置于后端的 Web 集群上。

(8) 支持虚拟主机。很多人认为它不支持虚拟主机，其实它是支持虚拟主机的。

注意 要证明 HAProxy 支持虚拟主机，我们可以在 HAProxy (192.168.1.6) 后端的 Nginx 配置两个虚拟主机，即 `www.atext.com` 和 `www.btest.com`，其他设定很简单，这里略过。`nginx.conf` 文件的新增内容如下：

```
server {
    listen      80;
    server_name www.atext.com;
    location / {
        root    /data/htdocs/www/atext;
        index   index.html index.htm;
    }
    error_page  500 502 503 504  /50x.html;
    location = /50x.html {
        root    html;
    }
}

server {
    listen      80;
    server_name www.btest.com;
    location / {
        root    /data/htdocs/www/btest;
        index   index.html index.htm;
    }
}
```



```

error_page 500 502 503 504 /50x.html;
location = /50x.html {
    root html;
}
}

```

然后在我们的 Windows XP 客户机上配置 c:\windows\system32\etc\drivers\hosts 文件，执行该操作的目的是让 hosts 暂时取代 DNS 指向。为了便于实验，添加如下内容：

```

192.168.1.6 www.atext.com
192.168.1.6 www.btest.com

```

最后我们在 Windows XP 客户机上的 IE 浏览器中输入 <http://www.atext.com> 和 <http://www.btest.com>，验证 HAProxy 是否支持虚拟主机。答案是肯定的。

6.1.6 高可用软件 Keepalived

Keepalived 是一款优秀的实现高可用的软件，它运行在 LVS 之上，它的主要功能是实现真实机的故障隔离及负载均衡器间的失败切换（failover）。Keepalived 是一个类似于 Layer3、Layer4、Layer5 交换机制的软件，也就是我们平时说的第3层、第4层和第5层交换。Keepalived 的作用是检测 Web 服务器的状态，如果有一台 Web 服务器死机，或者工作出现故障，Keepalived 将检测到，并将有故障的 Web 服务器从系统中剔除，当 Web 服务器工作正常后 Keepalived 会自动将 Web 服务器加入到服务器群中。这些工作全部自动完成，不需要人工干涉，需要人工做的只是修复故障的 Web 服务器。它的主要特点如下所示：

1) Keepalived 是 LVS 的扩展项目，因此它们之间具备良好的兼容性。这点应该是 Keepalived 部署比其他类似工具更简洁的原因，尤其是相对于 Heartbeat 而言，Heartbeat 作为 HA 软件，其复杂的配置流程让许多人望而生畏。

2) 通过对服务器池对象的健康检查，实现对失效机器/服务的故障隔离。

3) 负载均衡器之间的失败切换（failover），是通过 VRRPv2（Virtual Router Redundancy Protocol）stack 实现的，VRRP 当初被设计出来的目的就是为了解决静态路由器的单点故障问题。

4) 通过实际的线上项目，我们可以得知，iptables 的启用是不会影响 Keepalived 的运行的。但为了更好的性能，我们通常会将整套系统内所有主机的 iptables 都停用。

5) Keepalived 产生的 VIP 就是整个系统对外的 IP，如果最外端的防火墙采用的是路由模式，那就映射此内网 IP 为公网 IP。

Keepalived 是一款优秀的 HA 软件，我们现在多将其应于生产环境下的 LVS、Nginx 中，一般都是采取的双机方案，以保证网站最前端负载均衡器的高可用性。

6.1.7 高可用软件 Heartbeat

Linux-HA 的全称是 High-Availability Linux，它是一个开源项目。这个开源项目的目标是：通过社区开发者的共同努力，提供一个增强 Linux 可靠性（reliability）、可用性（availability）和可服务性（serviceability）（RAS）的集群解决方案。其中 Heartbeat 就是 Linux-HA 项目中的一个组件，也是目前开源 HA 项目中最成功的一个例子，它提供了所有 HA 软件所需要的基本功能，比如心跳检

测和资源接管、监测群集中的系统服务、在群集中的节点间转移共享 IP 地址的所有者等。自 1999 年开始使用到现在, Heartbeat 在行业内得到了广泛的应用, 也发行了很多的版本, 可以从 Linux-HA 的官方网站 <http://www.linux-ha.org> 上下载到 Heartbeat 的最新版本。尽管 Heartbeat 有许多优异的特性, 但它配置起来非常麻烦, 而且如果双机之间的心跳线出了问题, 就很容易形成“脑裂的问题”, 这也是目前制约其被大规模部署应用的原因。在生产环境下, Heartbeat 可以与 DRBD 一起应用于线上的高可用文件系统, 我们公司的许多相关项目已经稳定运行了好几年, 并且 MySQL 官方也推荐将其作为实现 MySQL 高可用的一种手段, 所以我建议大家掌握它的技术要点, 也可将其用于自己的项目或公司。

6.1.8 高可用块设备 DRBD

DRBD (Distributed Replicated Block Device) 是一种块设备, 可以用于高可用 (HA) 之中。它的功能类似于一个网络 RAID-1, 工作原理如图 6-3 所示。当你将数据写入本地文件系统时, 数据还将会被发送到网络中的另一台主机上, 并以相同的形式记录在一个文件系统中。本地 (主节点) 与远程主机 (备节点) 的数据可以保证实时同步。当本地系统出现故障时, 远程主机上还保留着一份相同的数据, 可以继续使用。在高可用 (HA) 中使用 DRBD 功能, 可以代替一个共享盘阵。因为数据同时存在于本地主机和远程主机上, 切换时, 远程主机只要使用它上面的那份备份数据就可以继续服务了。

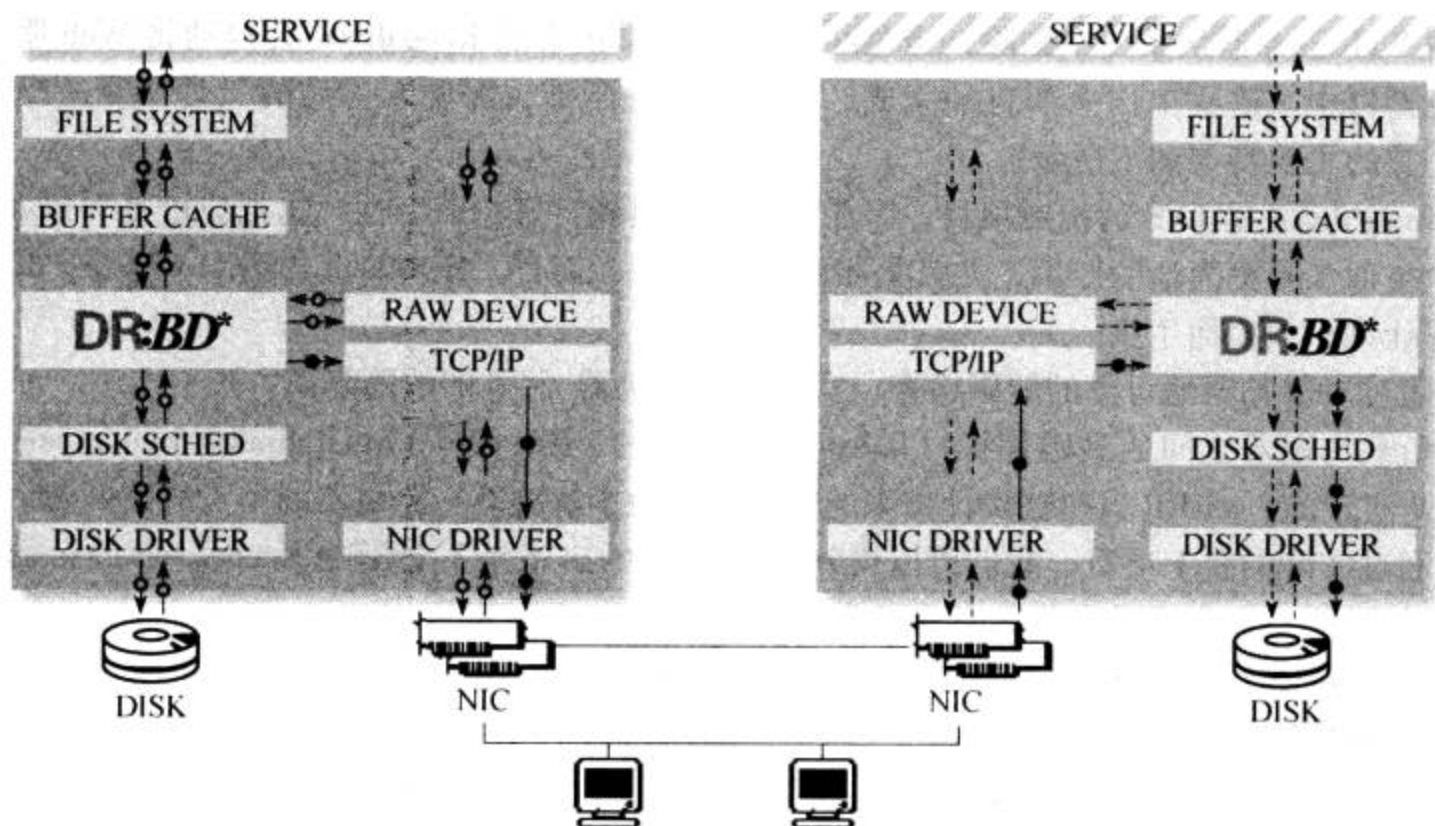


图 6-3 DRBD 工作原理图

DRBD 支持 3 种不同的复制协议, 允许 3 种程度的复制同步。

- 协议 A: 异步复制协议。只要主节点完成本地写操作就认为写操作完成, 并且需要复制的数据包会被存放到本地 TCP 发送缓存中。当发生 failover 故障, 数据可能会丢失。当 failover 故障发生时, 在 standby 节点的数据被认为仍是稳固的, 然而, 在 crash 发生的时间点上很多最新的更新操作会丢失。

- 协议 B: 内存同步 (半同步, semi-synchronous) 复制协议。当本地磁盘的写已经完成, 并且复制数据包已经到达对应从节点, 此时主节点才认为磁盘写已经完成。通常情况下, 发生 failover 不会导致数据丢失 (因为后备系统内存中已经获得了数据更新)。然而, 如果所有节点同时出现电源故障, 则主节点数据存储会发生不可逆的错误结构, 主节点上多数最新写入数据可能会丢失。
- 协议 C: 同步复制协议。只有在本地和远程磁盘都确定写入已完成时, 主节点才会认为写入完成。这样可确保发生单点故障时不会导致任何数据丢失。如果发生数据丢失的现象, 那也只会所有节点同时存在错误存储时才会发生这种情况。

在 DRBD 设置中, 最常用的复制协议是协议 C。选择哪种复制协议受部署的两个因素影响: 保护要求和延迟。我们在项目实施的过程中为了保证数据的一致性和可靠性, 均会选择协议 C。

另外, 我们在线上环境中主要是用 DRBD + Heartbeat + NFS 组成高可用的文件系统, 此项目上线两年多没有发生过丢失数据的现象。另外, DRBD 已被 MySQL 官方写入文档手册作为推荐的高可用方案之一。

6.2 负载均衡中的名词解释

6.2.1 什么是 Session

Session (会话) 是由应用服务器维持的一个服务器端的存储空间, 用户在连接服务器时, 会由服务器生成一个唯一的 SessionID, 该 SessionID 被作为标识符来存取服务器端的 Session 存储空间。

SessionID 这一数据是保存到客户端的, 用 Cookie 保存, 用户提交页面时, 会将这一 SessionID 提交到服务器端, 来存取 Session 数据。服务器也通过 URL 重写的方式来传递 SessionID 的值, 因此它不是完全依赖于 Cookie 的。如果客户端 Cookie 禁用, 则服务器可以自动通过重写 URL 的方式来保存 Session 的值, 并且这个过程对程序员透明。

6.2.2 什么是 Session 共享及实现的方法

当网站业务规模和访问量的逐步增大, 原本由单台服务器、单个域名组成的迷你网站架构可能已经无法满足发展需要。

此时我们可能会购买更多的服务器, 并且以频道化的方式启用多个二级子域名, 然后根据业务功能将网站分别部署在独立的服务器上, 或者通过负载均衡技术 (如 F5、LVS 等) 让多个频道共享一组服务器。

如果我们把网站程序分别部署到多台服务器上, 而且独立为几个二级域名, 由于 Session 存在实现原理上的局限性 (PHP 中 Session 默认以文件的形式保存在本地服务器的硬盘上), 这使得网站用户不得不经常在不同频道间来回输入用户名和密码登录, 导致用户体验大打折扣。另外, 原本程序可以直接从用户 Session 变量中读取的资料 (如: 昵称、积分、登录时间等), 因为无法跨服务器同步更新 Session 变量, 迫使开发人员必须实时读写数据库, 从而增加了数据库的负担。于是, 解决网站跨服务器的 Session 共享问题的需求变得迫切起来, 最终催生了多种解决方案。下面列举 3 种较为可行的方案来进行对比和探讨。

(1) 基于 Cookie 的 Session 共享

这个方案我们可能比较陌生，但它在大型网站中是被普遍使用了的。其原理是将全站用户的 Session 信息加密、序列化后以 Cookie 的方式统一种植在根域名下（如：.host.com）。当浏览器访问该根域名下的所有二级域名站点时，将与域名相对应的所有 Cookie 内容的特性传递给它，从而实现用户的 Cookie 化 Session 在多服务间的共享访问。

这个方案的优点是无需额外的服务器资源；缺点是由于受 HTTP 协议头信息长度的限制，仅能够存储小部分的个人信息，同时 Cookie 化的 Session 内容需要进行安全加解密（如：采用 DES、RSA 等进行明文加解密，再由 MD5、SHA-1 等算法进行防伪认证），另外它也会占用一定的带宽资源，因为浏览器会在请求当前域名下的任何资源时将本地 Cookie 附加在 http 头中传递到服务器上。

(2) 基于数据库的 Session 共享

首选当然是大名鼎鼎的 MySQL 数据库，并且建议使用内存表 Heap，以提高 Session 操作的读写效率。这个方案的实用性比较强，相信大家普遍在使用。它的缺点在于 Session 的并发读写能力取决于 MySQL 数据库的性能，同时需要我们自己来实现 Session 淘汰逻辑，以便定时从数据表中更新、删除 Session 记录，当并发过高时容易出现表锁，虽然我们可以选择行级锁的表引擎，但不得不否认使用数据库存储 Session 还是有杀鸡用牛刀之嫌。

(3) 基于 Memcache 的 Session 共享

Memcache 是一款基于 Libevent 的多路异步 I/O 技术的内存共享系统，简单的 Key + Value 数据存储模式使其代码逻辑小巧高效，因此在并发处理能力上占据了绝对优势。

另外值得一提的是 Memcache 的内存 Hash 表所特有的 Expires 数据过期淘汰机制，正好和 Session 的过期机制不谋而合，这就降低了删除过期 Session 数据的代码复杂度。但对比“基于数据库的存储方案”，仅逻辑这块就给数据表带来了巨大的查询压力。

事实上，我们在工作中也发现，以上方案中除了 Cookie 以外，其他两种方法均会给数据库带来巨大的查询压力。而我们的 MySQL 本身压力就非常大，这无疑是雪上加霜。其实我们应该在规划网站程序架构时就应该设计好，将程序集中放置在单台 Web 服务器上，并且用二级域名来保存海量的图片文件等，这样我们就可以通过 F5/LVS 的简单会话保持功能，以及 Nginx 的 IP_hash、HA-Proxy 的 balance source 机制来让用户始终与后端的一台 Web 服务器保持通信，这样 Session 共享的问题就很轻松地解决了。

6.2.3 什么是会话保持

在大多数的电子商务应用系统中，或者需要进行用户身份认证的在线系统中，一个客户与服务器经常会经过好几次的交互过程才能完成一笔交易或一个请求。由于这几次交互过程是密切相关的，服务器在进行这些交互的过程中，要完成某一个交互步骤往往需要了解上一次交互的处理结果，或者上几步的交互结果，这就要求所有相关的交互过程都由一台服务器完成，而不能被负载均衡器分散到不同的服务器上。

而这一系列相关的交互过程可能是由客户到服务器的一个连接的多次会话完成的，也可能是在客户与服务器之间的多个不同连接里的多次会话完成的。关于不同连接的多次会话，最典型的例子就是基于 HTTP 的访问，一个客户完成一笔交易可能需多次点击，而一个新的点击产生的请求，可能会重用上一次点击建立起来的连接，也可能是一个新建的连接。

会话保持就是指在负载均衡器上有这么一种机制，可以识别客户与服务器之间交互过程的关联性，在做负载均衡的同时，还能保证一系列相关联的访问请求被分配到同一台服务器上。

6.3 负载均衡器的会话保持机制

6.3.1 F5 Big-IP 的会话保持机制

F5 支持什么样的会话保持方法？

F5 Big-IP 支持多种会话保持方法，其中包括：简单会话保持（源地址会话保持）、HTTP Header 的会话保持、基于 SSL Session ID 的会话保持、I-Rules 会话保持，以及基于 HTTP Cookie 的会话保持，此外还有基于 SIP ID 及 Cache 设备的会话保持等，但常用的是简单会话保持、HTTP Header 的会话保持、HTTP Cookie 会话保持，以及基于 I-Rules 的会话保持。

1. 简单会话保持

简单会话保持也被称为基于源地址的会话保持，是指负载均衡器在做负载均衡时将访问请求的源地址作为判断关联会话的依据。在对来自同一 IP 地址的所有访问请求做负载均衡时会将其分配到同一台服务器上去。在 Big-IP 设备上可以通过网络掩码来区分“同一 IP 地址”，比如可以通过对 IP 地址 192.168.1.1 进行 255.255.255.0 的网络掩码，这样只要是来自于 192.168.1.0/24 这个网段的流量 Big-IP 都可以认为它们是来自于同一个用户的，这样就会把来自于 192.168.1.0/24 网段的流量会话保持到特定的一台服务器上。

简单会话保持里另外一个很重要的参数就是连接超时值，Big-IP 会为每一个进行会话保持的会话设定一个时间值，从某一个会话上一次完成到这个会话，下次再来之时的间隔小于这个超时值时，Big-IP 会将新的连接进行会话保持，但如果这个间隔大于该超时值，Big-IP 则会认为此连接是新的会话，然后进行负载均衡。

基于原地址的会话保持实现起来很简单，只需要根据数据包中三、四层的信息就可以实现，效率也比较高。它所存在的问题是，当多个客户是通过代理或地址转换的方式来访问服务器时，就会将其都分配到同一台服务器上，这就会导致服务器之间的负载严重失衡。另外一种情况是客户机数量很少，但每个客户机都会产生多个并发访问，对这些并发访问客户机要求通过负载均衡器分配到多个服务器上，这时基于客户端源地址的会话保持方法也会导致负载均衡失效。

2. 基于 Cookie 的会话保持

(1) Cookie 插入模式

在 Cookie 插入模式下，Big-IP 将负责插入 Cookie，后端服务器无须作出任何修改。当客户进行第一次请求时，客户 HTTP 请求（不带 Cookie）进入 Big-IP，Big-IP 根据负载均衡算法策略选择后端的一台服务器，并将请求发送至该服务器上，后端服务器进行 HTTP 回复（不带 Cookie）并发回至 Big-IP，然后 Big-IP 插入 Cookie，将 HTTP 回复返回到客户端。当客户请求再次发生时，客户 HTTP 请求（带有上次 Big-IP 插入的 Cookie）进入 Big-IP，然后 Big-IP 读出 Cookie 里的会话保持数值，将 HTTP 请求（带有与上面同样的 Cookie）发送到指定的服务器上，然后后端服务器进行请求回复，由于服务器并不写入 Cookie，HTTP 回复将不会带有 Cookie，恢复流量再次进入 Big-IP 时，Big-IP 再次写入更新后的会话保持 Cookie。

(2) Cookie 重写模式

在 Cookie 重写模式下，当客户进行第一次请求时，客户 HTTP 请求（不带 Cookie）进入 Big-IP，Big-IP 根据负载均衡算法策略选择后端的一台服务器，并将请求发送至该服务器上，后端服务器进行 HTTP 回复，并将一个空白的 Cookie 发回 Big-IP，然后 Big-IP 重新在 Cookie 里写入会话保持数值，再将 HTTP 回复返回到客户端。当客户请求再次发生时，客户 HTTP 请求（带有上次 Big-IP 重写的 Cookie）进入 Big-IP，然后 Big-IP 读出 Cookie 里的会话保持数值，将 HTTP 请求（带有与上面同样的 Cookie）发送到指定的服务器上，然后后端服务器进行请求回复。HTTP 回复里又将带有空的 Cookie，恢复流量再次进入 Big-IP 时，Big-IP 再次写入更新后会话保持数值到该 Cookie。

(3) Passive Cookie 模式

在 Passive Cookie 模式下，服务器使用特定信息来设置 Cookie。当客户进行第一次请求时，客户 HTTP 请求（不带 Cookie）进入 Big-IP，Big-IP 根据负载均衡算法策略选择后端的一台服务器，并将请求发送至该服务器上，后端服务器进行 HTTP 回复，并将一个 Cookie 发回 Big-IP。然后 Big-IP 将带有服务器写的 Cookie 值的 HTTP 回复返回到客户端。当客户请求再次发生时，客户 HTTP 请求（带有上次服务器写的 Cookie）进入 Big-IP，然后 Big-IP 根据 Cookie 里的会话保持数值，将 HTTP 请求（带有与上面同样的 Cookie）发送到指定的服务器上，接着后端服务器请求回复，HTTP 回复里又将带有更新的会话保持 Cookie，恢复流量再次进入 Big-IP 时，Big-IP 将带有该 Cookie 的请求回复给客户端。

(4) Cookie Hash 模式

当客户进行第一次请求时，客户 HTTP 请求（不带 Cookie）进入 Big-IP，Big-IP 根据负载均衡算法策略选择后端的一台服务器，并将请求发送至该服务器上，后端服务器进行 HTTP 回复，并将一个 Cookie 发回至 Big-IP 上。然后 Big-IP 将带有服务器写的 Cookie 值的 HTTP 回复返回到客户端。当客户请求再次发生时，客户 HTTP 请求（带有上次服务器写的 Cookie）进入 Big-IP，然后 Big-IP 根据 Cookie 里一定的某个字节的字节数来决定后台服务器是否接受请求，并将 HTTP 请求（带有与上面同样的 Cookie）发送到指定的服务器上。然后后端服务器进行请求回复，HTTP 回复里又将带有更新后的 Cookie，恢复流量再次进入 Big-IP 时，Big-IP 将带有该 Cookie 的请求回复给客户端。

3. SSL Session ID 会话保持

在用户的 SSL 访问系统的环境里，当 SSL 对话首次建立时，用户与服务器进行首次的信息交换：交换安全证书；商议加密和压缩方法；为每条对话建立 Session ID。由于该 Session ID 在系统中是一个唯一数值，因此，Big-IP 可以用该数值来进行会话保持。当用户想与该服务器再次建立连接时，Big-IP 可以通过会话中的 SSL Session ID 来识别该用户并进行会话保持。

基于 SSL Session ID 的会话保持就需要客户浏览器在进行会话的过程中始终保持其 SSL Session ID 不变，但实际上，我们发现微软的 Internet Explorer 在经过特定的时间后会主动改变 SSL Session ID，这就使基于 SSL Session ID 的会话保持实际应用范围大大缩小。

参考文档的地址如下所示：

<http://blog.csdn.net/laestev/article/details/2914930>

6.3.2 LVS 的会话保持机制

LVS 在 Linux 集群中是利用 persistence（单位为秒）来设定会话保持时间的，这个选项对于电子商务网站来说尤其有用：当用户从远程用账号登录网站时，有了这个会话保持功能，就能把用户

的请求转发给同一个应用服务器了。在这里，我们来做一个假设，假定现在有一个 LVS 环境，使用 VS/DR 转发模式，真实的 Web 服务器有两个，负载均衡器不启用会话保持功能。当用户第一次访问的时候，他的访问请求被负载均衡器转给某个真实服务器，这样他看到一个登录页面，第一次访问完毕；接着他在登录框里填写用户名和密码，然后提交；这时候，问题可能就会出现——登录不成功。因为没有会话保持，负载均衡器可能会把第 2 次的请求转发到其他的服务器上，这样浏览器又会提醒用户需要再次输入用户名及密码。

在这里我们可以做一个简单的实验来验证一下，实验的 IP 分配如表 6-1 所示。

表 6-1 LVS 会话实验的服务器 IP 分配表

服务器名称	IP	作用
LVS-MASTER	192.168.1.103	提供负载均衡
WEB1-REAL SERVER	192.168.1.106	提供 Web 服务
WEB2-REAL SERVER	192.168.1.107	提供 Web 服务
LVS-DR-VIP	192.168.1.108	集群的 VIP 地址

由于我这里是最小化安装，所以先安装编译工具等。另外为了不影响实验结果，强烈建议关闭防火墙和 SELinux，尤其是 iptables，它会直接影响实验结果。在后端的两台 Web 服务器上我直接安装了 httpd 服务，并分别设定了它们不同的首页地址，以示区分，如下所示：

```
yum -y install gcc gcc-c++ kernel-devel
```

1) 我们在 LVS-MASTER 上安装 LVS 软件，步骤如下：

```
#mkdir /usr/local/src/lvs
#cd /usr/local/src/lvs
#wget http://www.linuxvirtualserver.org/software/kernel-2.6/ipvsadm-1.24.tar.gz
#ln -s /usr/src/kernels/2.6.18-53.el5PAE-i686//usr/src/linux
```

上面这一步必须要进行，防止后面编译安装 ipvsadm 时找不到系统内核。

```
#tar zxvf ipvsadm-1.24.tar.gz
#cd ipvsadm-1.24
#make
#make install
```

2) 编写并运行 lvs.sh 脚本，绑定 VIP 地址到 LVS-MASTER 上，并设定 LVS 工作模式等。lvs.sh 脚本的代码内容如下所示：

```
#!/bin/bash
SNS_VIP=192.168.1.108
SNS_RIP1=192.168.1.106
SNS_RIP2=192.168.1.107

./etc/rc.d/init.d/functions

logger $0 called with $1
case "$1" in

    start)
        # set squid vip
        /sbin/ipvsadm -set 30 5 60
        /sbin/ifconfig eth0:0 $SNS_VIP broadcast $SNS_VIP netmask 255.255.255.255 broadcast $
SNS_VIP up
```



```

        /sbin/route add -host $SNS_VIP dev eth0:0
        /sbin/ipvsadm -A -t $SNS_VIP:80 -s wlc -p 120
        /sbin/ipvsadm -a -t $SNS_VIP:80 -r $SNS_RIP1:80 -g -w 1
        /sbin/ipvsadm -a -t $SNS_VIP:80 -r $SNS_RIP2:80 -g -w 1
        touch /var/lock/subsys/ipvsadm > /dev/null 2> &1

        ;;
stop)
        /sbin/ipvsadm -C
        /sbin/ipvsadm -Z
        ifconfig eth0:0 down
        route del $SNS_VIP
        rm -rf /var/lock/subsys/ipvsadm > /dev/null 2> &1
        echo "ipvsadm stoped"
        ;;

status)

        if [ ! -e /var/lock/subsys/ipvsadm ];then
                echo "ipvsadm stoped"
                exit 1
        else
                echo "ipvsadm OK"
        fi
        ;;

* )
        echo "Usage: $0 {start |stop |status}"
        exit 1
esac
exit 0

```

给脚本 x 权限，并执行它，命令如下所示：

```

chmod +x lvs.sh
./lvs.sh start

```

3) 在后端的两台 Web 服务器上执行 `realserver.sh` 脚本，此脚本的主要作用为：绑定 VIP 地址，并设定 arp 抑制。脚本 `realsearch.sh` 的代码如下所示：

```

#!/bin/bash
SNS_VIP=192.168.1.108
./etc/rc.d/init.d/functions

case "$1" in
start)
ifconfig lo:0 $SNS_VIP netmask 255.255.255.255 broadcast $SNS_VIP
/sbin/route add -host $SNS_VIP dev lo:0
echo "1" > /proc/sys/net/ipv4/conf/lo/arp_ignore
echo "2" > /proc/sys/net/ipv4/conf/lo/arp_announce
echo "1" > /proc/sys/net/ipv4/conf/all/arp_ignore

```



```

echo "2" > /proc/sys/net/ipv4/conf/all/arpvannounce
sysctl -p > /dev/null 2>&1
echo "RealServer Start OK"
;;
stop)
ifconfig lo:0 down
route del $LVS_VIP > /dev/null 2>&1
echo "0" > /proc/sys/net/ipv4/conf/lo/arp_ignore
echo "0" > /proc/sys/net/ipv4/conf/lo/arp_announce
echo "0" > /proc/sys/net/ipv4/conf/all/arp_ignore
echo "0" > /proc/sys/net/ipv4/conf/all/arp_announce
echo "RealServer Stopped"
;;
* )
echo "Usage: $0 {start|stop}"
exit 1
esac
exit 0

```

通过观察得知，当客户机 192.168.1.100 发起第一次连接请求时，LVS 负载均衡器将其分配到后面的真实物理服务器 192.168.1.104 上。在完成了三次握手后，其连接的状态为 ESTABLISHED（即 LVS 的 ActiveConn）。随后终止 TCP 连接，在终止 TCP 连接后相当长的一段时间内（约等于 15 分钟）192.168.1.100 再发起的新连接，都会一直连接到 192.168.1.107 上，也就是说 persistence 机制发挥了作用，这个时间我们可以用 `ipvsadm -t --timeout` 查看，即显示结果中的三项 tcp、tcpfin、udp 的 timeout 值。

1. Nginx 的会话保持机制

Nginx 中 upstream 模块的 ip_hash 机制能够将某个 IP 的请求定向到同一台后端服务器上，这样一来这个 IP 下的某个客户端和某个后端服务器就能建立起稳固的 Session 了，IP_hash 是在 upstream 配置中定义的：

```

upstream backend {
    ip_hash;
    server 192.168.1.106:80;
    server 192.168.1.107:80;
}

```

我们在线上采用了 Nginx + Keepalived 这种机制，而且采用这种机制的线上项目一直运行稳定，即使是在并发量大的情况下也没有发生过 Session 丢失的现象，这就证明了这种技术的可靠性，特推荐给大家。Nginx 现在在国内用得比较频繁，大家应该很熟悉其语法了，相关案例我会在后面介绍。

2. HAProxy 的会话保持机制

HAProxy 也有和 Nginx 的 IP_hash 类似的机制，即 balance source，它也可以实现会话保持功能。我们可以通过配置一个简单的 1+2 Web 架构来验证一下，此处的 IP 跟前面 LVS 中的一样，只不过这里没有 VIP 的概念了，详细过程如下。

1) 安装 HAProxy，步骤如下所示：

```

wget http://haproxy.lwt.eu/download/1.3/src/haproxy-1.3.20.tar.gz
tar zxvf haproxy-1.3.20.tar.gz

```

```
cd haproxy-1.3.20
make TARGET=linux26 PREFIX=/usr/local/haproxy
make install PREFIX=/usr/local/haproxy
cd /usr/local/haproxy
mkdir conf
```

2) 配置 HAProxy, 记得不要用它默认的轮询方式, 而是采用 `balance source`。配置文件 `/usr/local/haproxy/conf/haproxy.cfg` 的内容如下所示:

```
global
    log 127.0.0.1 local0
    maxconn 4096
    chroot /usr/local/haproxy
    uid 99
    gid 99
    daemon
    nbproc 1
    pidfile /usr/local/haproxy/logs/haproxy.pid
    debug

defaults
    log 127.0.0.1 local3
    mode http
    option httplog
    option httpclose
    option dontlognull
    option forwardfor
    option redispatch
    retries 2
    maxconn 2000
    balance source
    stats uri /haproxy-stats
    timeout 5000
    clitimeout 50000
    srvtimeout 50000

listen web_proxy 192.168.1.103:80
    #option httpchk HEAD /index.php HTTP/1.0
    server web1 192.168.1.106:80 cookie applinst1 check inter 2000 rise 2 fall 5
    server web2 192.168.1.107:80 cookie applinst2 check inter 2000 rise 2 fall 5
```

3) 以如下命令运行 HAProxy, 注意观察 HAProxy 的日志。命令如下所示:

```
/usr/local/haproxy/sbin/haproxy -f /usr/local/haproxy/conf/haproxy.cfg
```

HAProxy 正常运行的话应该有如下内容显示:

```
[WARNING] 125/001544 (6954) : <debug> mode incompatible with <quiet> and <daemon>. Keeping
<debug> only.
Available polling systems :
    sepoll : pref=400, test result OK
    epoll : pref=300, test result OK
```

```

poll : pref=200, test result OK
select : pref=150, test result OK
Total: 4 (4 usable), will use epoll.
Using epoll() as the polling mechanism.
00000000:web_proxy.accept(0004) = 0006 from [192.168.1.100:1233]
00000000:web_proxy.clireq[0006:ffff]: GET / HTTP/1.1
00000000:web_proxy.clidhr[0006:ffff]: Host: 192.168.1.103
00000000:web_proxy.clidhr[0006:ffff]: User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; zh-CN; rv:1.9.2.17) Gecko/20110420 Firefox/3.6.17

```

我们的客户端是 192.168.1.100，可以发现，无论怎么刷新始终都只是连接到 192.168.1.106 这个 Web 机器上，说明配置是成功的。如果改用 balance roundrobin 模式，那么客户端会轮流连接两台 Web 服务器。HAProxy 自带有强大的监控功能，监控页面如图 6-4 所示。

HAProxy version 1.3.20, released 2009/08/09

Statistics Report for pid 6954

> General process information

```

pid = 6954 process = 1, coproc = 1
uptime = 0s 00:00:00
system limits: memmax = unlimited, ulimit-s = 8192
maxsock = 8192 maxconn = 4096 maxpipe = 1
current conn = 1 current pipe = 1
Running since 1.3

```

```

active UP      backup UP
active UP going down  backup UP going down
active DOWN going up  backup DOWN going up
active or backup DOWN not checked
Note: UP mean load-balancing disabled is reported as "DOWN"

```

Display option:

- Hide DOWN servers
- Refresh now
- All servers

External resources:

- Frontend.log
- Backend.log
- Global log

web proxy	Queue			Session rate			Sessions				Bytes		Denied			Errors		Warnings		Server									
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	Left	In	Out	Req	Resp	Req	Conn	Resp	Rate	Ratio	Status	Weight	Act	Back	Chk	Down	Downtime	Thrott	
Frontend				1	-	-	1	1	2 000	22		9 025	3 772	0	0	0					OPEN								
web1	0	0	-	0	-	-	0	0	-	21	21	9 025	3 772	0	0	0	0	0	0	0	3m's UP	1	Y	-	0	0	0s	-	
web2	0	0	-	0	0	-	0	0	-	0	0	0	0	0	0	0	0	0	0	0	3m's UP	1	Y	-	0	0	0s	-	
Backend	0	0		0	0		0	1	2 000	21	21	9 025	3 772	0	0	0	0	0	0	0	3m's UP	2	2	0	0	0	0s		

图 6-4 HAProxy 的监控页面

下面介绍一下 HAProxy 的配置。

HAProxy 的配置中包含 5 个组件，如下所示（当然这些组件不是必选的，可以根据需要选择配置）。

- global：参数是进程级的，通常和操作系统（OS）相关。这些参数一般只设置一次，如果配置无误，就不需要再次配置了。
- defaults：配置默认参数，这些参数可以配置到 frontend、backend、listen 组件中。
- frontend：接收请求的前端虚拟节点，Frontend 可以根据规则直接指定具体使用后端的 backend（可动态选择）。
- backend：后端服务集群的配置，是真实的服务器，一个 backend 对应一个或多个实体服务器。
- listen：frontend 和 backend 的组合体。

下面是 HAProxy 的一些常用配置，这些配置可实现 HAProxy 的一些常用功能。具体配置请查看安装目录下 doc 目录下的文件。

配置的具体实例如下所示：

global

全局的日志配置，其中日志级别是 [err warning info debug]。

local0 是日志设备，必须为如下 24 种标准 syslog 设备中的一种：

- kern、user、mail、daemon、auth、syslog、lpr、news
- uucp、cron、auth2、ftp、ntp、audit、alert、cron2

□ local0、local1、local2、local3、local4、local5、local6、local7

由于之前在/etc/syslog.conf 文件中定义的是 local0，所以这里也用 local0。

```
log 127.0.0.1 local0 info #[err warning info debug]
```

以下为最大连接数：

```
maxconn 4096
```

用户如下：

```
user nobody
```

用户组如下：

```
group nobody
```

使 HAProxy 进程进入后台运行，以下是推荐的运行模式：

```
daemon
```

创建 4 个进程进入 daemon 模式运行。此参数要求将运行模式设置为 “daemon”。

```
nbproc 4
```

将所有进程的 pid 写入文件，启动进程的用户必须有访问此文件的权限。

```
pidfile /home/admin/haproxy/logs/haproxy.pid
```

```
defaults
```

默认的模式 mode|tcp|http|health|，tcp 是 4 层，http 是 7 层，health 只返回 OK。

```
mode http
```

采用 http 日志格式如下：

```
option httplog
```

3 次连接失败就认为是服务器不可用。也可以通过后面来设置。

```
retries 3
```

如果 Cookie 写入了 serverid 而客户端不会刷新 Cookie，待 serverid 对应的服务器挂掉后，将强制定向到其他健康的服务器上。

```
option redispatch
```

当服务器负载很高的时候，自动结束当前队列处理比较久的链接。

```
option abortonclose
```

默认的最大连接数如下：

```
maxconn 4096
```

连接超时如下：

```
timeout 5000
```

客户端超时如下：


```
clitimeout 30000
```

服务器超时如下：

```
srvtimeout 30000
```

心跳检测超时如下：

```
timeout check 1000
```

注意 一些参数值为时间，比如说 `timeout`。时间值通常的单位为毫秒 (ms)，但是也可以通过加#后缀来使用其他的单位。

下面是统计页面的配置：

```
listen admin_stats
```

监听端口如下：

```
bind 0.0.0.0:1080
```

http 的 7 层模式如下：

```
mode http
```

日志设置如下：

```
log 127.0.0.1 local0 err #[err warning info debug]
```

统计页面自动刷新时间如下：

```
stats refresh 30s
```

统计页面的 URL 如下：

```
stats uri /admin_stats
```

统计页面密码框上的提示文本如下：

```
stats realm Gemini\ Haproxy
```

统计页面用户名和密码设置如下：

```
stats auth admin:admin101
```

隐藏统计页面上 HAProxy 的版本信息如下：

```
stats hide-version
```

下面是网站检测的 `listen` 定义：

网站健康检测 URL，用来检测 HAProxy 管理的网站是否可用，它是依靠检查后端 Web 服务器是否存在 `index.php` 来判断后端主机是否挂掉的。如果后端的所有 Web 机器上均没有 `index.php` 或都挂掉了，那么我们访问 HAProxy 主机地址，例如 `http://192.168.1.103` 时，浏览器就会返回 503 Service Unavailable, No server is available to handle this request. 这行报错信息。

```
listen web_proxy 192.168.1.103:80
```

关于监听的其他配置选项 HAProxy 本身默认已经做好了，不需要太多人为的干预。

下面是 frontend 配置：

```
frontend http_80_in
```

监听端口如下：

```
bind 0.0.0.0:80
```

http 的 7 层模式如下：

```
mode http
```

应用全局的日志配置如下：

```
log global
```

启用 http 的 log 如下：

```
option httplog
```

每次请求完毕后主动关闭 http 通道，HA-Proxy 不支持 keep-alive 模式。

```
option httpclose
```

如果后端服务器需要获得客户端的真实 IP 则需要配置此参数，以便从 Http Header 中获得客户端 IP。

```
option forwardfor
```

下面是 HAProxy 的日志记录内容配置：

```
capture request header Host len 40
capture request header Content-Length len 10
capture request header Referer len 200
capture response header Server len 40
capture response header Content-Length len 10
capture response header Cache-Control len 8
```

下面是 acl 的策略定义。

如果请求的域名满足正则表达式则返回 true，-i 表示忽略大小写。

```
acl denali_policy hdr_reg (host) -i ^(www.gemini.taobao.net|my.gemini.taobao.net|auction1.
gemini.taobao.net) $
```

如果请求域名满足 trade.gemini.taobao.net 则返回 true，-i 表示忽略大小写。

```
acl tm_policy hdr_dom (host) -i trade.gemini.taobao.net
```

如果在请求 URL 中包含 sip_apiname = ，则此控制策略返回 true，否则为 false。

```
acl invalid_req URL_sub -i sip_apiname =
```

如果在请求 URL 中存在 timetask 作为部分地址路径，则此控制策略返回 true，否则返回 false。

```
acl timetask_req URL_dir -i timetask
```

当请求的 header 中 Content-length 等于 0 时返回 true。

```
acl missing_cl hdr_cnt (Content-length) eq 0
```

下面是与 acl 策略匹配的相应配置。

当请求的 header 中 Content-length 等于 0 时阻止请求并返回 403。

```
block if missing_cl
```

block 表示阻止请求，返回 403 错误。如果不满足策略 invalid_req，或者满足策略 timetask_req，则阻止请求。

```
block if ! invalid_req || timetask_req
```

当满足 denali_policy 的策略时使用 denali_server 的 backend。

```
use_backend denali_server if denali_policy
```

当满足 tm_policy 的策略时使用 tm_server 的 backend。

```
use_backend tm_server if tm_policy
```

reqisetbe 关键字定义，根据定义的关键字选择 backend。

```
reqisetbe ^Host:\img dynamic
reqisetbe ^[\^]*\/(img|css)/ dynamic
reqisetbe ^[\^]*\ /admin/stats stats
```

以上都不满足的时候使用默认 mms_server 的 backend。

```
default_backend mms_server
```

HAProxy 的错误页面设置如下所示：

```
errorfile 400 /home/admin/haproxy/errorfiles/400.http
errorfile 403 /home/admin/haproxy/errorfiles/403.http
errorfile 408 /home/admin/haproxy/errorfiles/408.http
errorfile 500 /home/admin/haproxy/errorfiles/500.http
errorfile 502 /home/admin/haproxy/errorfiles/502.http
errorfile 503 /home/admin/haproxy/errorfiles/503.http
errorfile 504 /home/admin/haproxy/errorfiles/504.http
```

下面是 backend 的设置：

```
backend mms_server
```

http 的 7 层模式如下：

```
mode http
```

负载均衡的方式，roundrobin 平均方式如下：

```
balance roundrobin
```

允许插入 serverid 到 Cookie 中，serverid 在后面可以定义如下：

```
cookie SERVERID
```

心跳检测的 URL，HTTP/1.1 和 Host: XXXX 指定了心跳检测 HTTP 的版本，XXXX 为检测时请求服务器的 request 中的域名是什么。如果在应用的检测 URL 对应的功能中有对域名依赖的话则需

要设置如下：

```
option httpchk GET /member/login.jhtml HTTP/1.1\r\nHost:member1.gemini.taobao.net
```

服务器定义如下，cookie 1 表示 serverid 为 1，check inter 1500 是检测心跳频率，rise 3 是 3 次正确认为服务器可用，fall 3 是 3 次失败认为服务器不可用，weight 代表权重。

```
server mms1 10.1.5.134:80 cookie 1 check inter 1500 rise 3 fall 3 weight 1
server mms2 10.1.6.118:80 cookie 2 check inter 1500 rise 3 fall 3 weight 2
backend denali_server
mode http
```

负载均衡的方式，source 根据客户端 IP 进行哈希的方式（类似 Nginx 的 ip_hash 机制）。

```
balance source
```

如果设置了 backup，默认第一个 backup 会优先，设置 option allbackups 后所有备份服务器的权重一样。

```
option allbackups
```

心跳检测 URL 设置如下：

```
option httpchk GET /mytaobao/home/my_taobao.jhtml HTTP/1.1\r\nHost:my.gemini.taobao.net
```

可以根据机器性能的不同，不使用默认的连接数配置而使用自己的特殊连接数配置，比如 minconn 10 maxconn 20。

```
server denlai1 10.1.5.114:80 minconn 4 maxconn 12 check inter 1500 rise 3 fall 3
server denlai2 10.1.6.104:80 minconn 10 maxconn 20 check inter 1500 rise 3 fall 3
```

备份机的配置如下。正常情况下备份机不会使用，当主机的全部服务器都宕机的时候备份机才会被启用。

```
server dnali-back1 10.1.7.114:80 check backup inter 1500 rise 3 fall 3
server dnali-back2 10.1.7.114:80 check backup inter 1500 rise 3 fall 3
backend tm_server
mode http
```

负载均衡的方式，leastconn 根据服务器当前的请求数，取当前请求数最少的服务器。

```
balance leastconn
option httpchk GET /trade/itemlist/prepayCard.htm HTTP/1.1\r\nHost:trade.gemini.taobao.net
server tm1 10.1.5.115:80 check inter 1500 rise 3 fall 3
server tm2 10.1.6.105:80 check inter 1500 rise 3 fall 3
```

下面是 reqisetbe 自定义的关键字匹配 backend 的部分。

```
backend dynamic
mode http
balance source
option httpchk GET /welcome.html HTTP/1.1\r\nHost:www.taobao.net
server denlai1 10.3.5.114:80 check inter 1500 rise 3 fall 3
server denlai2 10.4.6.104:80 check inter 1500 rise 3 fall 3
backend stats
```



```

mode http
balance source
option httpchk GET /welcome.html HTTP/1.1\r\n Host:www.163.com
server denlai1 10.5.5.114:80 check inter 1500 rise 3 fall 3
server denlai2 10.6.6.104:80 check inter 1500 rise 3 fall 3

```

参考文档:

<http://haproxy.1wt.eu/download/1.4/doc/configuration.txt>

<http://code.google.com/p/haproxy-docs/>

HAProxy 官方网站: <http://haproxy.1wt.eu/>

相对而言, HAProxy 的正则较之 Nginx 复杂一些, 但我们也可以参考上面的详细 HAProxy 配置文档和参考文档来写。HAProxy 实际配置文件非常简单, 大家可以参考《HAProxy 的会话保持机制》这篇, 没有必要全部应用以上选项。

在项目实施中, 我会根据客户的需求, 将 HAProxy 用于一些时效性强的小型网站上 (比如证券类的新闻资讯网站), 做成单 HAProxy1+2Web 的 Web 资讯网站, 因为这些网站只是在早上9:00以后到下午5:00之间会有用户访问。鉴于 HAProxy 的稳定性、接近硬件设备的网络吞吐量, 以及其所拥有的强大监控功能, 完全可以胜任这项工作。如果大家也有这种需求, 不妨考虑一下这种 1+2 型的做法。另外, HAProxy + Heartbeat 目前也只是用于测试环境, 暂时没有将其用于线上环境, 在稍后的案例中我会详细介绍。

6.4 Linux 集群的项目案例分享

6.4.1 项目案例一: 用 Nginx + Keepalived 实现在线票务系统

这是一套以前实施过的在线订票系统, 防火墙、交换机、服务器均置于电信机房的同一机柜中, 服务器的带宽 40MB, 项目拓扑图如图 6-5 所示。

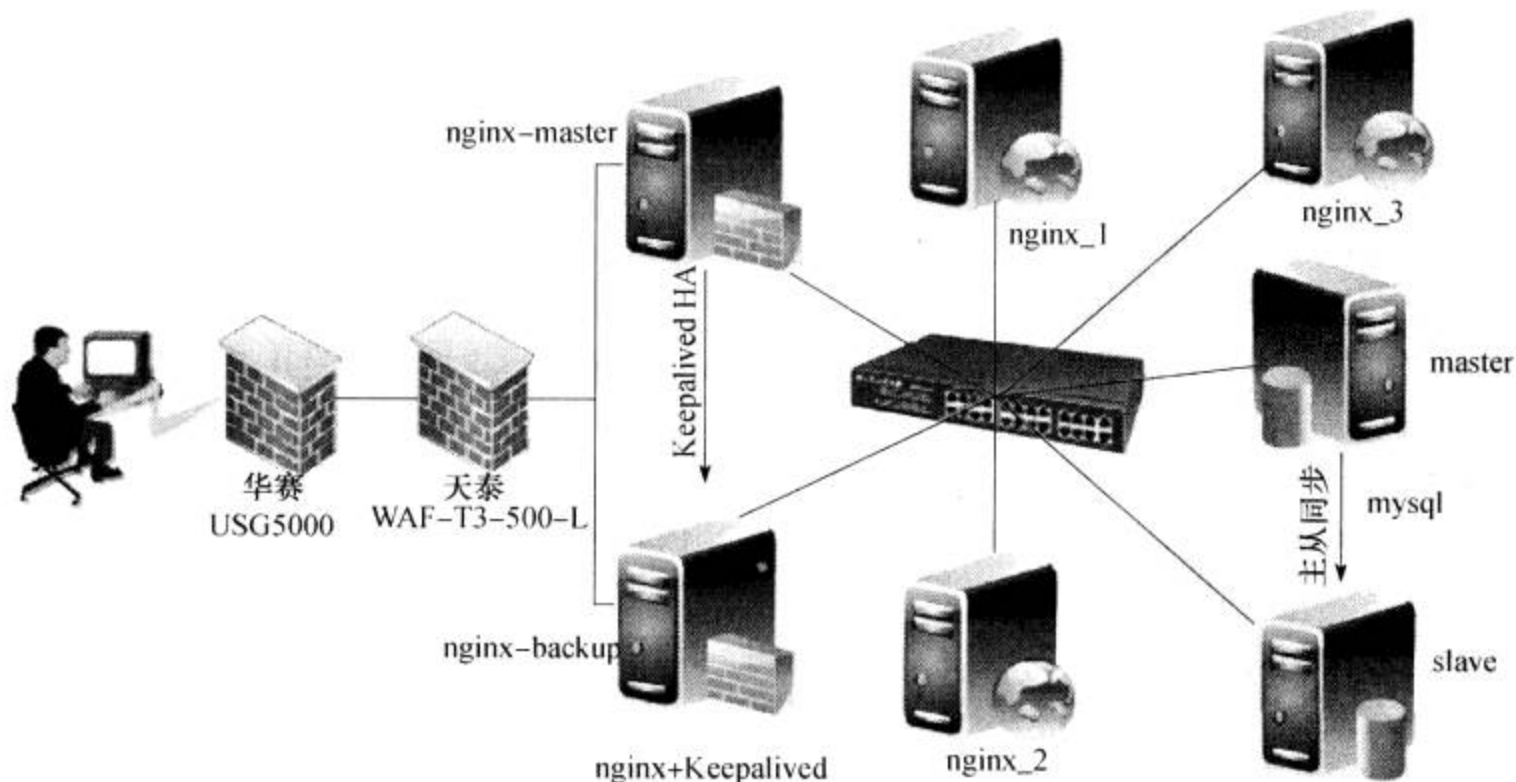


图 6-5 在线订票系统网络拓扑图

1. 整套系统的安全考虑

因为牵涉到信用卡和银行卡在线付款的问题，所以整套系统的安全性就必须注意了。在项目实施过程中，安全防护这块我们是采用 3 层硬件防火墙及 7 层应用层防护来实现的，系统均安装了 64 位的 Centos5.4。软件层分负载均衡层、Web 层、数据库层，整套系统均关闭 iptables 防火墙，只映射 Keepalived 虚拟的 VIP 在最前端的华赛 USG5000 的外网 80 端口上，先将整套系统的安全级别提高到金融安全级别，再考虑负载均衡及其他事宜。另外，网络工程师都应该清楚，防火墙有 3 种工作模式，路由、透明和混合模式。华赛 USG5000 防火墙用的是路由模式，而天泰防火墙用的则是透明模式。

这里稍微介绍一下这两种防火墙。

华赛 USG5000 可以有效抵御高强度的网络攻击，同时还可以保证正常的网络应用。其基于多核处理器的硬件构架，依靠多线程处理设计提供了十分优异的数据处理能力，完全可以为 Internet 服务提供商、大型企业、园区网、数据中心等具备大流量网络带宽的用户提供高性能的安全防御手段。尤其是 USG5000 所具备的超高的“每秒新建连接数”，不仅可以对多种并行的网络应用实现快速响应，而且在大流量网络攻击的情况下，仍然可以防止网络业务中断，避免给用户带来损失。

它能有效地保障网络的运行，并且其独特的 GTP 安全防护功能可以为 GPRS 网络提供有效的安全防护。USG5000 安全网关可以抵御大流量的 DDoS 攻击，为用户的业务系统提供 DDoS 攻击防护，依托其优越的产品性能，能防范每秒数百万包以上的 DDoS 攻击，可对 SYN FLOOD、UDP FLOOD、ICMP FLOOD、DNS FLOOD、CC 等多种 DDoS 攻击种类准确识别并控制，同时还能提供蠕虫病毒流量的识别和防范能力，结合华为赛门铁克专有的 ICA 智能连接算法，保证在准确识别 DDoS 攻击流量的同时，不影响用户的正常访问，在复杂的网络情况下实现真正的安全防护，是业界领先的 DDoS 防护设备。这功能也是我们关注的重点。

天泰 WAF-T3-500-L 安全网关防火墙具备全面的攻击防御系统，可保证系统不受网络蠕虫病毒和应用专用漏洞的攻击，并且大大缓解了来自网络层和应用层 DoS/DDoS 攻击的影响。网关的 Net-Shield 引擎在网络层会对数据进行细致检查，彻底阻断来自网络层的潜在攻击。网关的 WebShield 引擎会在应用层对 Web 请求进行检查，辨别恶意内容并阻止其进入应用服务器。

其安全性能如下。

- 常见网络攻击防护：保护网络基础设施不受常见的来自网络层的攻击。
- DoS/DDoS 保护：识别网络层和应用层的 DoS/DDoS 攻击，缓解攻击对应用基础设施的影响（这也是我们关注的重点）。
- 入侵过滤：通过在恶意蠕虫和病毒进入应用服务器前进行识别并拒绝其进入，保护应用服务器不受侵袭。
- SSL 加密：应用内容在传输过程中都受加密保护，通过转移服务器复杂的加/解密任务将应用处理能力发挥到极致。该功能保护敏感应用内容的安全，使其摆脱被窃取及被滥用的潜在威胁。

此外，它能实现 SQL 注入攻击防护、钓鱼攻击的防护、跨站脚本攻击的防护以及常见系统溢出的防护等，这也是我们比较关注的内容。

关于证书，我们购买的是商业证书，建议大家关注 Geo Trust 的证书，它是支持多域名的，价格比较高。另外，我们还购买了 McAfee 的网站扫描服务，这一服务是针对代码安全的。

2. 硬件方面的投入

服务器我们一般采用采用的是 HP DL380G6（用于后面的 Web 集群）和 HP DL580G5（用于 MySQL 数据库）。在项目实施过程中我们发现 HP DL580G5 的性能确实彪悍，如果资金充分，可考虑采用此服务器作为应用服务器。它跟以往的老型号不同，用的是双四核至强 E74403.2G 的 CPU，内存一般是 64GB 或 146GB，这可以根据项目成本来权衡。在租用机房时，我们一般选择的是电信机房，也可考虑北京双线通机房，出口带宽建议为 40MB（如果资金确实宽裕，也可以考虑 100MB）。

3. 负载均衡层

负载均衡层的软件我们采用的是 Nginx0.8.15 源码（当时最新最稳定的版本），两台 Nginx 负载均衡用 Keepalived 作高 HA。其实也可以用 LVS/Keepalived 来实现，但我们在项目实施过程中发现，Nginx 在正则处理及分发上效果比 LVS 更好（有些功能 LVS 实现不了），而且稳定性也不错。在已上线的金融资讯类的项目里，我做的是 1+2 的架构（按客户的要求），已经稳定运行几年了。当然也要配合 Cacti + Nagios 进实时监控。

如何处理 Session 的问题呢？

- 1) Nginx 负载均衡器采用 ip_hash 模块，让访问的客户端始终与后端的某台 Web 服务器建立永久性的连接关系（即保持会话功能）。
- 2) 采用与 PHPCMS 类似的方法，将 Session 写进后端的统一数据库里，例如 MySQL。
- 3) 利用 memcached 统一管理 Session。

前期我们用的是第二种方案，后来发现数据库的压力会因此而增大，所以采取了前一种会话保持的方法。

4. Web 层

页面同步的办法如下：

- 1) 不同 Web 服务器之间数据的同步我们可以采用 rsync 的办法。
- 2) 后端采用共用存储，读数据采用同一个存储设备。

这里说一下要用的存储设备，我们用得比较多的是 EMC CLARiiON CX4 的 FC 磁盘阵列，它很稳定，没发生过丢失数据库的问题。其缺点是比较贵，会增加整个系统的实施成本。

- 3) 在 PHP 程序上实现动态的调取数据（如图片信息），不在 Web 服务器调取而直接采用后端的文件服务器或存储。

前期我们采用的是单 NFS 方案，后期采用的是 DRBD + Heartbeat + NFS 方案（有的客户要求要上存储，我们就上了 EMC CLARiiON CX4），其稳定性还是很不错的。

Web 集群方面用的是 Nginx + PHP5 (FastCGI)。这里说一下并发的的问题。在设计项目方案时，我们考虑单台 Web 服务器上的并发值为 3000，在局域网环境中通过 LoadRunner 及 Webbench 反复测试，单台 Nginx 的 Web 服务器通过 3000 的并发没有问题，3 台 Web 即是 $3000 \times 3 = 9000$ 并发。但系统正式上线时发现，在非游戏类的网站上根本达不到 9000 并发，这只是一个理论值。但本着高扩展性的原则，还是尽量在硬件和性能上对单台 Web 服务器进行了调优。我们要设计的这个票务系统是 9000 万张票，预计并发在 2000 左右，此系统架构完全能胜任此并发情况。另外 Nginx 作为负载均衡器/代理服务器在高并发下的稳定性是毋庸置疑的，有相关项目经验的人都应该很清楚这点。

5. 数据库层

考虑到数据库层的压力情况，我提出 4 种设计方案：

射出来的外网地址。

本节内容出自我的项目方案，这种负载均衡方式同时也应用于我公司的电子商务网站中，目前已稳定上线一年多了。通过下面的内容，大家可以迅速架构一个企业级的负载均衡高可用的Web环境。在负载均衡高可用技术上，我一直主力推崇以Nginx + Keepalived作Web的负载均衡高可用架构，并积极将其应用于真实项目中，此架构极适合灵活稳定的环境。Nginx负载均衡作服务器遇到的故障如下：

- 服务器网线松动等网络故障。
- 服务器硬件故障发生损坏现象而崩溃。
- Nginx服务死掉。

遇到前两种情况时，Keepalived是能起到HA的作用的；然而遇到第3种情况时就没有办法了，但可以通过SHELL监控解决这个问题，从而实现真正意义上的负载均衡高可用。我在电子商务网站上就采用了这种方法，下面详细说明一下其安装步骤。

1. Nginx + Keepalived 的说明及环境说明

关于服务器系统，从早期的Centos5.1 x86_64到现在的Centos5.5 x86_64我们均有涉及，有时应客户的要求，也有大量RHEL5.Xx86_64系列的主机。整个系统的IP情况如表6-2所示。

表 6-2 Nginx + Keepalived 服务器的 IP 分配表

服务器名称	IP	作用
Nginx_MASTER	192.168.1.103	提供负载均衡
Nginx_BACKUP	192.168.1.104	负载均衡备机
Nginx_VIP_TP	192.168.1.108	网站的VIP地址
WEB 1_REAL SERVER	192.168.1.106	提供Web服务
WEB 2_REAL SERVER	192.168.1.107	提供Web服务

2. 分别安装 Nginx 负载均衡器及相关的配置脚本

先安装Nginx负载均衡器，Nginx负载的设置就用一般的模板来配置了。

1) 添加运行Nginx的用户和组www及Nginx存放日志的位置，并安装gcc等基础库，以免发生libtool报错现象，命令如下：

```
yum -y install gcc gcc + gcc - c ++ openssl openssl - devel
groupadd www
useradd -g www www
mkdir -p /data/logs/
chown -R www:www /data/logs/
```

2) 下载并安装Nginx0.8.15（当时最新最稳定的版本，现在我推荐用Nginx0.8.56）。另外建议工作中养成好习惯，所下载的软件包均放到/usr/local/src下，如下所示：

```
cd /usr/local/src
wget http://blog.s135.com/soft/linux/nginx_php/pcr/pcr-7.9.tar.gz
tar zxvf pcr-7.9.tar.gz
cd pcr-7.9/
./configure
make && make install
cd ../
wget http://sysoev.ru/nginx/nginx-0.8.15.tar.gz
tar zxvf nginx-0.8.15.tar.gz
cd nginx-0.8.15/
```

```
./configure --user=www --group=www --prefix=/usr/local/nginx --with-http_stub_status_
module --with-http_ssl_module
make && make install
cd ../
```

配置 Nginx 负载均衡器的配置文件是 `vim /usr/local/nginx/conf/nginx.conf`，下面的配置文件仅仅是我某项目的配置文档，纯 http 转发。如果要添加 SSL 支持也简单，后面会有相关说明。一定要将购买的相关证书文件放到 Nginx 负载均衡器上（两台均要放），而非置于后面的 Web 机器上。配置文件内容如下：

```
user www www;
worker_processes 8;
pid /usr/local/nginx/logs/nginx.pid;
worker_rlimit_nofile 51200;
events
{
    use epoll;
    worker_connections 51200;
}
http{
    include mime.types;
    default_type application/octet-stream;
    server_names_hash_bucket_size 128;
    client_header_buffer_size 32k;
    large_client_header_buffers 4 32k;
    client_max_body_size 8m;
    sendfile on;
    tcp_nopush on;
    keepalive_timeout 60;
    tcp_nodelay on;
    fastcgi_connect_timeout 300;
    fastcgi_send_timeout 300;
    fastcgi_read_timeout 300;
    fastcgi_buffer_size 64k;
    fastcgi_buffers 4 64k;
    fastcgi_busy_buffers_size 128k;
    fastcgi_temp_file_write_size 128k;
    gzip on;
    gzip_min_length 1k;
    gzip_buffers 4 16k;
    gzip_http_version 1.0;
    gzip_comp_level 2;
    gzip_types text/plain application/x-javascript text/css application/xml;
    gzip_vary on;

    upstream backend
    {
        ip_hash;
        server 192.168.1.106:80;
        server 192.168.1.107:80;
    }
```

```

server {
listen 80;
server_name www.1paituan.com;
location / {
root /var/www/html ;
index index.php index.htm index.html;
proxy_redirect off;
proxy_set_header Host $host;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For proxy_add_x_forwarded_for;
proxy_pass http://backend;
}

location /nginx {
access_log off;
stub_status on;
#auth_basic_user_file /usr/local/nginx/htpasswd;
}

log_format access 'remote_addr - remote_user [time_local] "request" '
'$statusbody_bytes_sent "$http_referer" '
'"$http_user_agent"$http_x_forwarded_for';
access_log /data/logs/access.log access;
}
}

```

分别在两台 Nginx 负载均衡器上执行 `/usr/local/nginx/sbin/nginx` 命令，启动 Nginx 进程，然后用 `lsof -i: 80` 命令来检查，命令如下：

```
lsof -i:80
```

此命令显示结果如下：

COMMAND	PID	USER	FD	TYPE	DEVICE	SIZE	NODE	NAME
nginx	13875	root	6u	IPv4	25918		TCP *	:http (LISTEN)
nginx	13876	www	6u	IPv4	25918		TCP *	:http (LISTEN)
nginx	13877	www	6u	IPv4	25918		TCP *	:http (LISTEN)
nginx	13878	www	6u	IPv4	25918		TCP *	:http (LISTEN)
nginx	13879	www	6u	IPv4	25918		TCP *	:http (LISTEN)
nginx	13880	www	6u	IPv4	25918		TCP *	:http (LISTEN)
nginx	13881	www	6u	IPv4	25918		TCP *	:http (LISTEN)
nginx	13882	www	6u	IPv4	25918		TCP *	:http (LISTEN)
nginx	13883	www	6u	IPv4	25918		TCP *	:http (LISTEN)

Nginx 程序正常启动后，两台 Nginx 负载均衡器就算安装成功了，现在我们要安装 Keepalived 来实现这两台 Nginx 负载均衡器的高可用。

3. 安装 Keepalived，让其分别作 Web 及 Nginx 的 HA

1) 安装 Keepalived，并将其做成服务模式，方便以后调试。Keepalived 的安装方法如下：

```

wget http://www.keepalived.org/software/keepalived-1.1.15.tar.gz
tar zxvf keepalived-1.1.15.tar.gz

```

```
cd keepalived-1.1.15
./configure --prefix=/usr/local/keepalived
make
make install
```

2) 安装成功后做成服务模式, 方便启动和关闭, 方法如下:

```
cp /usr/local/keepalived/sbin/keepalived /usr/sbin/
cp /usr/local/keepalived/etc/sysconfig/keepalived /etc/sysconfig/
cp /usr/local/keepalived/etc/rc.d/init.d/keepalived /etc/init.d/
```

3) 分别设置主和备 Nginx 上的 Keepalived 配置文件。我们先配置主 Nginx 上的 keepalived.conf 文件, 如下所示:

```
#mkdir /etc/keepalived
#cd /etc/keepalived/
```

我们可以用 vim 编辑 /etc/keepalived/keepalived.conf, 内容如下所示:

```
! Configuration File for keepalived
global_defs {
    notification_email {
        yuhongchun027@163.com
    }
    notification_email_from keepalived@chtopnet.com
    smtp_server 127.0.0.1
    smtp_connect_timeout 30
    router_id LVS_DEVEL
}
vrrp_instance VI_1 {
    state MASTER
    interface eth0
    virtual_router_id 51
    mcast_src_ip 192.168.1.103
    # 此处是主 Nginx 的 IP 地址, 此机的 priority 为 100
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass chtopnet
    }
    virtual_ipaddress {
        192.168.1.108
    }
}
```

下面设置备用 Nginx 上的 keepalived.conf 配置文件, 注意与主 Nginx 上的 keepalived.conf 区分开, 代码如下:

```
! Configuration File for keepalived
global_defs {
    notification_email {
```



```

yuhongchun027@163.com
}
notification_email_from keepalived@chtopnet.com
smtp_server 127.0.0.1
smtp_connect_timeout 30
router_id LVS_DEVEL
}
vrrp_instance VI_1 {
    state MASTER
    interface eth0
    virtual_router_id 51
    mcast_src_ip 192.168.1.104
    priority 99
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass chtopnet
    }
    virtual_ipaddress {
        192.168.1.108
    }
}

```

在两台负载均衡器上分别启动 Keepalived 程序，命令如下：

```
service keepalived start
```

我们来看一下日志，这是主 Nginx 上与 Keepalived 相关的日志，我们可以用如下命令查看：

```
tail /var/log/messages
```

此命令显示结果如下：

```

May 6 05:10:42 localhost Keepalived_vrrp: Configuration is using : 62610 Bytes
May 6 05:10:42 localhost Keepalived_vrrp: VRRP sockpool: [ ifindex(2),proto(112),fd(8,9) ]
May 6 05:10:43 localhost Keepalived_vrrp: VRRP_Instance(VI_1) Transition to MASTER STATE
May 6 05:10:44 localhost Keepalived_vrrp: VRRP_Instance(VI_1) Entering MASTER STATE
May 6 05:10:44 localhost Keepalived_vrrp: VRRP_Instance(VI_1) setting protocol VIPs.
May 6 05:10:44 localhost Keepalived_healthcheckers: Netlink reflector reports IP 192.168.1.108 added
May 6 05:10:44 localhost Keepalived_vrrp: VRRP_Instance(VI_1) Sending gratuitous ARPs on eth0 for 192.168.1.108
May 6 05:10:44 localhost Keepalived_vrrp: Netlink reflector reports IP 192.168.1.108 added
May 6 05:10:44 localhost avahi-daemon[2212]: Registering new address record for 192.168.1.108 on eth0.
May 6 05:10:49 localhost Keepalived_vrrp: VRRP_Instance(VI_1) Sending gratuitous ARPs on eth0 for 192.168.1.108

```

很显然 VRRP（虚拟路由冗余协议）已经启动。我们还可以通过命令 `ip addr` 来检查主 Nginx 上的 IP 分配情况，通过下面的显示内容可以清楚地看到 VIP 地址已绑定到主 Nginx 的机器上了。

```

ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue

```

```

link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
  inet 127.0.0.1/8 scope host lo
  inet6 ::1/128 scope host
    valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
  link/ether 00:0c:29:51:59:df brd ff:ff:ff:ff:ff:ff
  inet 192.168.1.103/24 brd 192.168.1.255 scope global eth0
  inet 192.168.1.108/32 scope global eth0
  inet6 fe80::20c:29ff:fe51:59df/64 scope link
    valid_lft forever preferred_lft forever
3: sit0: <NOARP> mtu 1480 qdisc noop
  link/sit 0.0.0.0 brd 0.0.0.0

```

在这个过程中，有大量的 VRRP 数据包。那什么是 VRRP 呢？为了让大家对 Keepalived 有所了解，这里我详细向大家说明 VRRP（虚拟路由冗余协议）。

随着 Internet 的迅猛发展，基于网络的应用逐渐增多。这就对网络的可靠性提出了越来越高的要求。斥资对所有网络设备进行更新当然是一种很好的可靠性解决方案，但从保护现有资产的角度考虑，可以采用廉价冗余的思路，在可靠性和经济性方面找到平衡点。

虚拟路由冗余协议就是一种很好的解决方案。在该协议中，对共享多存取访问介质（如以太网）上的终端 IP 设备的默认网关（Default Gateway）进行冗余备份，当其中一台路由设备宕机时，备份路由设备及时接管转发工作，向用户提供透明的切换，提高了网络服务质量。

（1）协议概述

在基于 TCP/IP 协议的网络中，为了保证不直接物理连接的设备之间的通信，必须指定路由。目前常用的指定路由的方法有两种：一种是通过路由协议（比如：内部路由协议 RIP 和 OSPF）动态学习；另一种是静态配置。在每一个终端都运行动态路由协议是不现实的，大多客户端操作系统平台都不支持动态路由协议，即使支持也受到管理开销、收敛度、安全性等许多问题的限制。因此普遍采用对终端 IP 设备进行静态路由配置的方式，一般是给终端设备指定一个或多个默认网关（Default Gateway）。静态路由的方法简化了网络管理的复杂度，也减轻了终端设备的通信开销。但是它有一个缺点：如果作为默认网关的路由器损坏，所有使用该网关为下一跳主机的通信必然要中断。即便配置了多个默认网关，如不重新启动终端设备，也不能切换到新的网关上。但是采用虚拟路由冗余协议就可以很好地避免静态指定网关的缺陷。

在 VRRP 协议中，有两组重要的概念：VRRP 路由器和虚拟路由器，主控路由器和备份路由器。VRRP 路由器是指运行 VRRP 的路由器，是物理实体，虚拟路由器是指 VRRP 协议创建的逻辑概念。一组 VRRP 路由器协同工作，共同构成一台虚拟路由器。该虚拟路由器对外表现为一个具有唯一固定 IP 地址和 MAC 地址的逻辑路由器。处于同一个 VRRP 组中的路由器具有两种互斥的角色：主控路由器和备份路由器，一个 VRRP 组中有且只有一台处于主控角色的路由器，可以有一个或多个处于备份角色的路由器。VRRP 协议使用选择策略从路由器组中选出一台作为主控，负责 ARP 响应和转发 IP 数据包，组中的其他路由器作为备份的角色处于待命状态。当由于某种原因主控路由器发生故障时，备份路由器能在几秒钟的延时后升级为主路由器。由于此切换非常迅速而且不用改变 IP 地址和 MAC 地址，故对终端使用者的系统是透明的。

（2）工作原理

一个 VRRP 路由器有唯一的标识：VRID，范围为 0 ~ 255。该路由器对外表现为唯一的虚拟

MAC 地址，地址的格式为 00-00-5E-00-01-[VRID]。主控路由器负责对 ARP 请求用该 MAC 地址做应答。这样，无论如何切换，保证给终端设备的是唯一一致的 IP 和 MAC 地址，减少了切换对终端设备的影响。

VRRP 控制报文只有一种：VRRP 通告 (advertisement)。它使用 IP 多播数据包进行封装，组地址为 224.0.0.18，发布范围只限于同一局域网内。这保证了 VRID 在不同的网络中可以重复使用。为了减少网络带宽消耗，只有主控路由器才可以周期性地发送 VRRP 通告报文。备份路由器在连续 3 个通告间隔内收不到 VRRP 或收到优先级为 0 的通告则启动新一轮 VRRP 选举。

在 VRRP 路由器组中，按优先级选举主控路由器，VRRP 协议中的优先级范围是 0~255。若 VRRP 路由器的 IP 地址和虚拟路由器的接口 IP 地址相同，则称该虚拟路由器为 VRRP 组中的 IP 地址所有者。IP 地址所有者自动具有最高优先级：255。优先级 0 一般用在 IP 地址所有者主动放弃主控者角色时。可配置的优先级范围为 1~254。优先级的配置原则可以依据链路的速度、成本、路由器的性能和可靠性，以及其他管理策略来设定。在主控路由器的选举中，高优先级的虚拟路由器会获胜，因此，如果在 VRRP 组中有 IP 地址所有者，则它总是以主控路由器的角色出现。对于有相同优先级的候选路由器，则按照 IP 地址的大小顺序选举。VRRP 还提供了优先级抢占策略，如果配置了该策略，高优先级的备份路由器便会剥夺当前低优先级的主控路由器的权力而成为新的主控路由器。

为了保证 VRRP 协议的安全性，提供了两种安全认证措施：明文认证和 IP 头认证。明文认证方式要求：在加入一个 VRRP 路由器组时，必须同时提供相同的 VRID 和明文密码，可避免在局域网内的配置错误，但不能防止通过网络监听方式获得密码。IP 头认证的方式提供了更高的安全性，能够防止报文重放和修改等攻击。

我们可以通过 tcpdump 抓包发现两台 Nginx 负载均衡器上是否有大量 VRRP 包。可以在任何一台机器上开启 tcpdump 进行抓包，命令如下：

```
tcpdump vrrp
```

命令显示结果如下：

```
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
18:04:16.372116 IP 192.168.1.103 > VRRP.MCAST.NET: VRRPv2, Advertisement, vrid 51, prio 100, auth-
type simple, interval 1s, length 20
18:04:17.374134 IP 192.168.1.103 > VRRP.MCAST.NET: VRRPv2, Advertisement, vrid 51, prio 100, auth-
type simple, interval 1s, length 20
18:04:18.375461 IP 192.168.1.103 > VRRP.MCAST.NET: VRRPv2, Advertisement, vrid 51, prio 100, auth-
type simple, interval 1s, length 20
18:04:19.376198 IP 192.168.1.103 > VRRP.MCAST.NET: VRRPv2, Advertisement, vrid 51, prio 100, auth-
type simple, interval 1s, length 20
18:04:20.377229 IP 192.168.1.103 > VRRP.MCAST.NET: VRRPv2, Advertisement, vrid 51, prio 100, auth-
type simple, interval 1s, length 20
18:04:20.378986 IP 192.168.1.104 > VRRP.MCAST.NET: VRRPv2, Advertisement, vrid 51, prio 99, auth-
type simple, interval 1s, length 20
18:04:20.381515 IP 192.168.1.103 > VRRP.MCAST.NET: VRRPv2, Advertisement, vrid 51, prio 100, auth-
type simple, interval 1s, length 20
18:04:21.383936 IP 192.168.1.103 > VRRP.MCAST.NET: VRRPv2, Advertisement, vrid 51, prio 100, auth-
type simple, interval 1s, length 20
```


可以看到，优先级高的一方（即 100），通过 VRRPv2，获得了 VIP 地址；而且 VRRPv2 包的发送极有规律，每秒发送一次。当然了，这是通过配置文件进行控制的。

4. 用 Nginx_pid.sh 来监控 Nginx 进程，实现真正意义上的负载均衡高可用

针对 Nginx + Keepalived 方案，编写了 Nginx 监控脚本 nginx_pid.sh。此脚本的思路其实也很简单，即将其放置在后台一直监控 Nginx 进程，如果进程消失，则尝试重启 Nginx；如果失败，则立即停掉本机的 Keepalived 服务，让另一台负载均衡器接手。此脚本直接从生产环境下载，内容如下所示：

```
#!/bin/bash
while :
do
nginxpid='ps -C nginx --no-header | wc -l'
if [ $nginxpid -eq 0 ];then
/usr/local/nginx/sbin/nginx
sleep 5
nginxpid='ps -C nginx --no-header | wc -l'
echo $nginxpid
if [ $nginxpid -eq 0 ];then
/etc/init.d/keepalived stop
fi
fi
sleep 5
done
```

然后将其置于后台运行 `sh /root/nginx_pid.sh &`，做到这步时大家要注意一下，这种写法是有问题的，我们用 root 用户退出终端后，此进程便会消失，正确写法为 `nohup /bin/bash /root/nginx_pid.sh &`。

说明 nohup 的作用：如果你正在运行一个进程，而且你觉得在退出账户时该进程还不应结束，那么可以使用 nohup 命令，该命令可以在你退出 root 账户之后继续运行相应的进程。nohup 就是不挂起的意思（no hang up）。

这个脚本也取自我公司目前电子商务网站的线上服务器，在几次线上维护工作中，我们手动人为重启了主 Nginx 服务器，在非常短的时间内就从 Nginx 切换过来了，用户没有因为网站故障而进行投诉，事实证明此脚本还是有效的。

5. 模拟故障测试

整套系统配置完成后，我们就可以通过 `http://192.168.1.108/` 来访问了。我们曾做过一些模拟性故障测试，比如关掉一台 Nginx 负载均衡器，抽掉某台 Web 机器的网线或直接关机，甚至停掉其中一台 Nginx 服务器的 Nginx 服务。我们会发现无论在什么情况下，Nginx + Keepalived 都可以正常提供服务。我的许多项目（包括电子商务网站）都是用此架构的，并且至少在线上稳定运行一年以上（有的已经运行几年了）。

6. Nginx 作为负载均衡器在工作中遇到的问题

1) 如何让 Nginx 负载均衡器也支持 HTTPs?

要让 Nginx 支持 HTTPs，方法其实很简单，在负载均衡器上开启 SSL 功能，监听 443 端口（防火墙上也要做好映射），将证书放在 Nginx 负载均衡器上（不是后面的 Web 服务器），就可轻松解

决此问题。详见以下 nginx.conf 配置文件，代码如下：

```
server
{
    listen 443;
    server_name www.cn7788.com;

    ssl on;
    ssl_certificate /usr/local/nginx/keys/www.cn7788.com.crt;
    ssl_certificate_key /usr/local/nginx/keys/www.cn7788.com.key;

    ssl_protocols SSLv3 TLSv1;
    ssl_ciphers ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:-LOW:-SSLv2:-EXP;
}
```

2) 如何让后端的 Apache 服务器获取客户端的真实 IP?

运行在后端 Apache 服务器上的应用所获取到的 IP 都是 Nginx 负载均衡所在服务器的 IP，或者是本机 127.0.0.1。你查看 Apache 的访问日志，就会见到来来去去的都是内网的 IP。虽然可以通过 Nginx 日志来判断客户端的 IP，但有些考虑不周全的应用，例如 Tattertools（一个博客程序）就会犯错，在后台的访问日志上总显示访客数 1，IP 来自 127.0.0.1。这时候就要想办法来处理了。其实我们可以通过修改 Nginx proxy 的参数令后端应用获取到 Nginx 发来的请求报文，并获取外网的 IP，在 Nginx 的配置文件中记得加上如下内容：

```
proxy_set_header    Host $host;
proxy_set_header    X-Real-IP $remote_addr;
proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
```

这仅仅是让 Nginx 获取外网 IP，但 Apache 未必买账呢！也就是说 Apache 端同样需要设置，搜寻了一下，发现了 Apache 有一个来自第三方的 module（模块）配合 Nginx proxy 使用。下面简要说一下这个模块。

模块相关说明：<http://stderr.net/apache/rpaf/>

模块的下载地址：<http://stderr.net/apache/rpaf/download/>

最新版本 mod_rpaf-0.6.tar.gz

安装也相当简单。

```
tar zxvf mod_rpaf-0.6.tar.gz
```

下载后解压。

```
cd mod_rpaf-0.6
```

Apache 的目录按自己的环境修改，并选择相应的安装方式，命令如下：

```
/usr/local/apache/bin/apxs -i -c -n mod_rpaf-2.0.so mod_rpaf-2.0.c
```

完成后会在 http.conf 的 LoadModule 区域多加了一行内容，如下所示：

```
LoadModule mod_rpaf-2.0.so modules/mod_rpaf-2.0.so
```

经 Apache 2.2.6 的实验证明，使用这一行启动 Apache 的时候会报错。

所以将其修改如下：

```
LoadModule rpaf_module          modules/mod_rpaf-2.0.so
```

并在下方添加如下内容：

```
RPAFenable On
RPAFsethostname On
RPAFproxy_ips 127.0.0.1 192.168.1.101 192.168.102
RPAFheader X-Forwarded-For
```

在填写 Nginx 所在的内网 IP 时，Nginx 的内网地址是必写的，不然一样会失败。另外，有几个代理服务器的 IP 就写几个代理服务器的 IP。

保存退出后重启 Apache，再看看 Apache 的日志内容，应该已经很完美地解决这个问题了。

3) 正确区分 Nginx 的分发请求。

我在设计某个小项目时，原本是基于 Nginx 的 1 + 3 架构，开发人员突然要增加一台机器（Windows Server 2003 系统），专门存放图片及 PDF 等，项目要求能在 Nginx 后的 3 台 Web 机器上显示图片，并且可以下载 pdf。当时我有点懵，因为程序用的是 Zend Framework，所以一直在用正则表达式作为跳转。后来才想明白其中的“玄机”，IE 程序先在 Nginx 负载均衡器上提申请，所以 nginx.conf 是在做分发而非正则跳转，此时最前端的 Nginx，既是负载均衡器也是反向代理。明白这个就好办了。另外要注意 location/StockInfo 与 location ~ ^/StockInfo 的差异性，Nginx 默认是正则优先的。顺便也说一下，proxy_pass 支持直接写 IP 的方式。Nginx 的部分代码如下：

```
upstream mysrv {
    ip_hash;
    server 192.168.110.62;
    server 192.168.110.63;
}

upstream myjpg {
    server 192.168.110.3:88;
}

server
{
    listen 80;
    server_name web.tfzq.com;
    proxy_redirect off;

    location ~ ^/StockInfo{
        proxy_pass http://myjpg;
    }
}
```

4) 单 Nginx 负载均衡器的监控。

有时因为项目的适时性，比如许多资讯类的小网站，客户只需要单 Nginx 负载均衡器，这个时候我们该如何监控呢？其实方法也很简单，如果网站比较重要，我们可以购买 Alertbot，它会每秒扫描你的网站一次，遇到问题会即时发邮件给你。不过，有了它也别大意，Naiogs 监控一定要打开，Alertbot 扫描你的网站是基于公网的，它发出警报时不一定是网站出了问题，很有可能是防火

墙出问题了。如果是 Nagios 和 Alertbot 同时报警，那一定要慎重，这种情况肯定是网站的服务器出问题了，此时我们应该将注意力集中到服务器上而非防火墙上。

总结：目前此套架构在 2000 并发的电子商务网站上运行非常稳定，唯一不足之处就是 Nginx_backup 一直处于闲置状态。相对于双主 Nginx 负载均衡器而言，此架构比较简单，出问题的几率相对也较小，而且出问题时容易排障，在网站收录方面需要考虑的问题也非常少，所以我一直采用这种方案。通过线上的观察，我们不难发现，Nginx 作负载均衡器/反向代理也是相当稳定的，可以媲美硬件级的 F5 了，我想这也是越来越多的朋友喜欢它的原因之一。Nginx + Keepalived 用于生产环境的优势还是很多的，不过由于其自身的限制，它目前只应用于 Web 集群环境。这种 Web 级负载均衡高可用架构我目前正在通过 www.51cto.com 和 www.chinaunix.net 社区推广，如果大家的网站或项目有这种需求，不妨考虑应用一下。

6.4.3 项目案例三：用 LVS + Keepalived 构建高可用 JSP 集群

在此项目案例中，整个网站的服务器均置于武汉电信机房内，整个项目的拓扑图如图 6-7 所示。

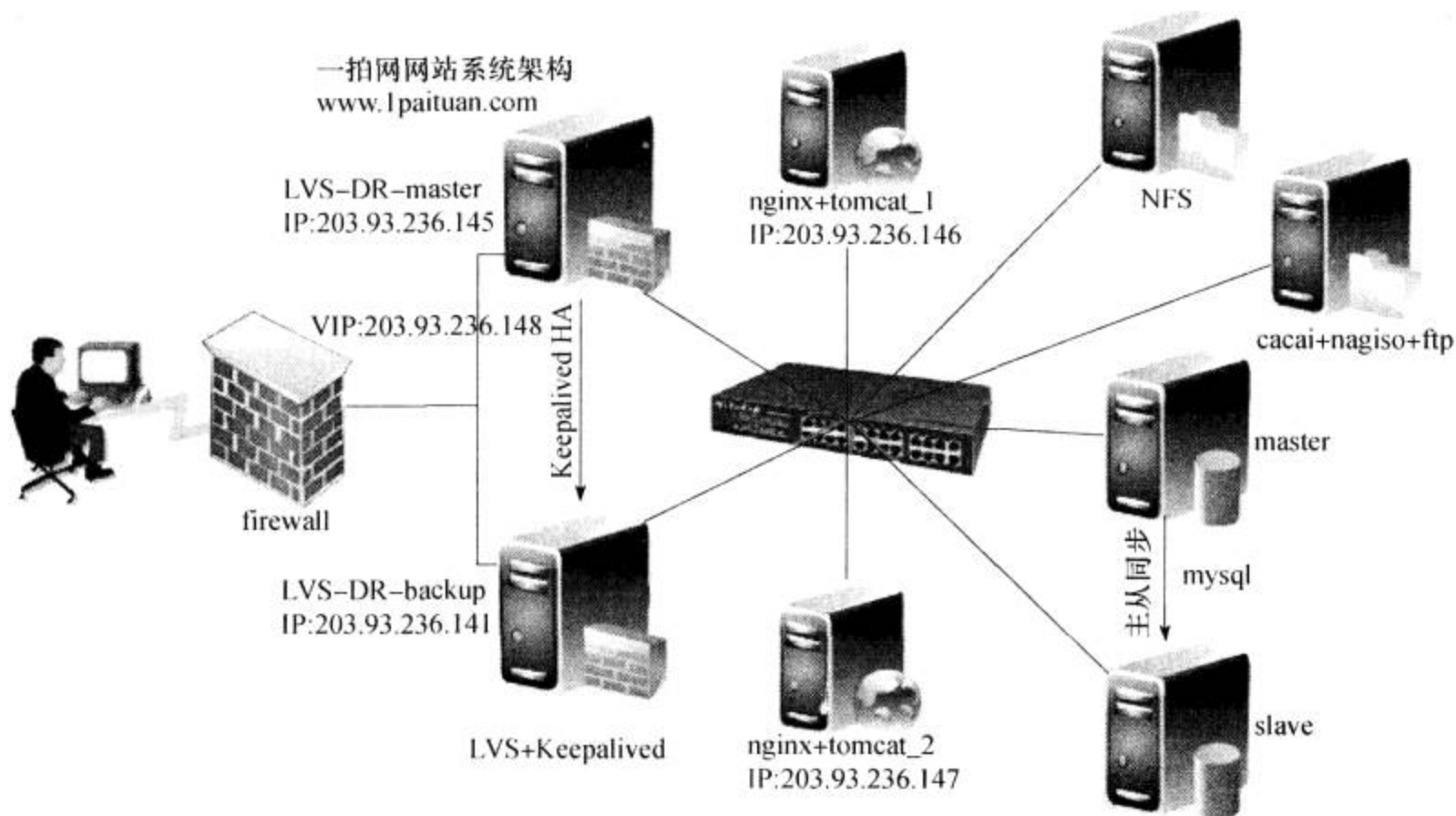


图 6-7 一拍网网站系统架构图

主要服务器涉及的公网 IP 如表 6-3 所示。

表 6-3 一拍网网站服务器的 IP 分配表

服务器名称	IP	服务器用途
LVS-MASTER	203.93.236.145	主 LVS 负载均衡器
LVS-BACKUP	203.93.236.141	备 LVS 负载均衡器
LVS-VIP-IP	203.93.236.148	同时也是网站的公网 IP
WEB 1_REAL SERVER	203.93.236.146	提供 Web 服务及 JSP 服务
WEB 2_REAL SERVER	203.93.236.147	提供 Web 服务及 JSP 服务

Tomcat 是 Apache 软件基金会 (Apache Software Foundation) 中 Jakarta 项目的一个核心项目, 由 Apache、Sun 和其他一些公司及个人共同开发而成。由于有了 Sun 的参与和支持, 最新的 Servlet 和 JSP 规范总是能在 Tomcat 中得到体现, Tomcat 5 支持最新的 Servlet 2.4 和 JSP 2.0 规范。因为 Tomcat 技术先进、性能稳定, 而且免费, 因此深受 Java 爱好者的喜爱并得到了部分软件开发商的认可, 成为了目前比较流行的 Web 应用服务器。目前最新版本是 7.0。

Tomcat 7.0 相对于以前的版本来说, 它的新特征如下:

- ☐ 使用随机数防止跨站脚本攻击。
- ☐ 改变了安全认证中的 jessionid 的机制, 防止 Session 攻击。
- ☐ 可进行内存泄露的侦测和防范。
- ☐ 在 war 文件外使用别名存储静态内容。
- ☐ 支持 Servlet 3.0、JSP 2.2 和 JSP-EL 2.2。
- ☐ 更容易将 Tomcat 内嵌到应用中去, 比如 JBoss。
- ☐ 异步日志。

其中第三点是我最为关注的, 在以前的版本中 Tomcat 均存在着内存泄露的情况, 我希望能在新版本中有所改善。

Nginx 与 Tomcat 整合的好处如下:

- ☐ 静态分离, 加快用户访问网站的速度。
- ☐ Tomcat 处理 JSP 还行, 但处理标志文件效率就太差了。
- ☐ 整个负载均衡层和 Web 层的工作流程为 LVS/DR + Keepalived → Nginx 负载均衡层 → Tomcat 集群, 可以保证整个网站不会因为某一台 LVS 或 Nginx 机器挂掉而影响网站的运营。
- ☐ Nginx 稳定, 宕机的可能性微乎其乎。

下面是项目实施的具体步骤。

由于服务器均采用的是最小化安装, 所以先安装一些基础的编译库, 命令如下:

```
yum -y install gcc gcc-c++ autoconf libjpeg libjpeg-devel libpng libpng-devel freetype freetype-devel libxml2 libxml2-devel zlib zlib-devel glibc glibc-devel glib2 glib2-devel bzip2 bzip2-devel ncurses ncurses-devel cURL cURL-devel e2fsprogs e2fsprogs-devel krb5 krb5-devel libidn libidn-devel openssl openssl-devel
```

1) 分别在 203.93.236.146 和 203.93.236.147 上安装 Nginx 0.8.56 和 Tomcat 7.0, 然后整合它们。

在安装 Tomcat 7.0 之前, 必须先安装 JDK, 其下载地址为 https://cds.sun.com/is-bin/INTERSHOP.enfinity/WFS/CDS-CDS_Developer-Site/en_US/-/USD/ViewFilteredProducts-SingleVariationType-Filter, 由于我的服务器都是 Centos x86_64, 所以选择了 jdk-6u18-linux-x64.bin 软件包。记得还是要将其放在 /usr/local/src 目录下, 方便以后的工作规划。

下载完成后, 修改 jdk-6u18-linux-x64.bin 为可执行, 并运行它。

```
cd /usr/local/src
chmod +x jdk-6u18-linux-x64.bin
./jdk-6u18-linux-x64.bin
mv jdk1.6.0_18 /usr/local/jdk
```

接着配置系统的 Java 运行环境, 我是通过修改 /etc/profile 文件来实现, 修改内容如下所示:

```
JAVA_HOME="/usr/local/jdk"
```



```
CLASS_PATH="$JAVA_HOME/lib:$JAVA_HOME/jre/lib"
PATH=".: $PATH: $JAVA_HOME/bin"
CATALINA_HOME="/usr/local/tomcat"
export JAVA_HOME CATALINA_HOME
```

保存退出后，执行以下命令让环境立即生效：

```
source /etc/profile
```

然后下载并安装 apache-tomcat7.0.12，如下所示：

```
cd /usr/local/src/
wget http://mirror.bjtu.edu.cn/apache/tomcat/tomcat-7/v7.0.12/bin/apache-tomcat-7.0.12.tar.gz
tar xzvf apache-tomcat-7.0.12.tar.gz
mv apache-tomcat-7.0.12 /usr/local/tomcat/
cp -rf /usr/local/tomcat/webapps /data/htdocs/www
vim /usr/local/tomcat/conf/server.xml
```

修改 Tomcat 的根路径位置，我的网站地址为 /data/htdocs/www/shop，这个虚拟主机需要在 /usr/local/tomcat/conf/server.xml 里指定。改动后的内容如下所示：

```
<Host name="www.lpaituan.com" appBase="webapps"
      unpackWARs="true" autoDeploy="true">
  <Context path="/" docBase="/data/htdocs/www/shop/" />
```

如果我们要继续增加虚拟主机，按照如上格式继续添加内容即可。Host name 后面接虚拟主机名称，docBase 后面接虚拟主机对应的路径位置。

安装完成后，启动 Tomcat，默认即监听了 8080 端口。启动命令如下：

```
/usr/local/tomcat/bin/startup.sh
```

成功运行后，用 lsof -i:80 来进行验证，如下所示：

```
lsof -i:8080
COMMAND  PID USER  FD  TYPE  DEVICE SIZE NODE NAME
java    23731 root   40u  IPv6 3347645    TCP *:webcache (LISTEN)
```

最后安装 pcre8.1.0 及 Nginx0.8.56，整合 Nginx0.8.56 与 Tomcat7.0.12，Nginx0.8.56 的安装过程可参考前面的内容，为了节约篇幅就不再详细说明了。静态 HTML 页面、图片、CSS 等由 Nginx 来处理，jsp、do 内容由后端的 Tomcat 处理，nginx.conf 配置文件的内容如下所示（这里为了调试方便，我们首先将域名 www.lpaituan.com 指向了 203.93.236.146，另一台 Web 配置跟 203.93.236.146 一样，我这里以 203.93.236.146 举例说明，等此架构中的 Web 环境均顺利后，再将域名 www.lpaituan.com 指向我们的 VIP 地址）：

```
user www www;
worker_processes 8;
error_log /usr/local/webserver/nginx/logs/nginx_error.log crit;
pid /usr/local/webserver/nginx/nginx.pid;
#Specifies the value for maximum file descriptors that can be opened by this process.
worker_rlimit_nofile 65535;
#工作模式及连接数上限
```

```

events
{
use epoll;
worker_connections 65535;
}
#设定 http 服务器,利用它的反向代理功能提供负载均衡支持
http
{
    #设定 mime 类型
    include      mime.types;
    default_type application/octet-stream;
    #charset  gb2312;
    #设定请求缓冲
    server_names_hash_bucket_size 128;
    client_header_buffer_size 32k;
    large_client_header_buffers 4 32k;
    client_max_body_size 300m;
    sendfile on;
    tcp_nopush    on;
    keepalive_timeout 60;
    tcp_nodelay on;
    server_tokens off;
    client_body_buffer_size 512k;
    proxy_connect_timeout 5;
    proxy_send_timeout 60;
    proxy_read_timeout 5;
    proxy_buffer_size 16k;
    proxy_buffers 4 64k;
    proxy_busy_buffers_size 128k;
    proxy_temp_file_write_size 128k;
    # fastcgi_connect_timeout 300;
    # fastcgi_send_timeout 300;
    # fastcgi_read_timeout 300;
    # fastcgi_buffer_size 64k;
    # fastcgi_buffers 4 64k;
    # fastcgi_busy_buffers_size 128k;
    # fastcgi_temp_file_write_size 128k;
    gzip on;
    gzip_min_length 1k;
    gzip_buffers 4 16k;
    gzip_http_version 1.1;
    gzip_comp_level 2;
    gzip_types      text/plain application/x-javascript text/css application/xml;
    gzip_vary on;

    #limit_zone crawler $binary_remote_addr 10m;

    ###禁止通过 IP 访问站点
    server{
        server_name _;

```

```

return 404;
}
server
{
listen      80;
server_name  www.lpaituan.com;
index index.html index.htm index.jsp index.do; #设定访问的默认首页地址
root  /data/htdocs/www/shop; #设定网站的资源存放路径
#limit_conn  crawler 20;
if (-d $request_filename)
{
rewrite ^/(.*) ([^/])$ http://$host/$1$2/ permanent;
}
#所有 jsp 的页面均交由 Tomcat 处理
location ~ \.(jsp|jspx|do)?$ {
proxy_set_header    Host $host;
proxy_set_header    X-Real-IP $remote_addr;
proxy_pass http://127.0.0.1:8080; #转向 Tomcat 处理
}
location ~ .*\. (htm|html|gif|jpg|jpeg|png|bmp|swf|ioc|rar|zip|txt|flv|mid|doc|ppt|pdf|xls|mp3|wma) $ #设定访问静态文件直接读取不经过 Tomcat
{
expires      30d;
}
location ~ .*\. (js|css)?$
{
expires      1h;
}

#定义访问日志的写入格式
log_format  wwwlog  '$remote_addr - $remote_user [ $time_local] "$request" '
' $status $body_bytes_sent "$http_referer" '
'" $http_user_agent" $http_x_forwarded_for';
access_log  /data/logs/www/nginx.log wwwlog; #设定访问日志的存放路径
}
}

```

配置完成后，我们用如下命令启动 Nginx 进程，让其监听 80 端口，如下所示：

```
/usr/local/webserver/nginx/sbin/nginx
```

然后我们在 /data/htdocs/www/shop/ 下面写一段 Java 代码，验证以上的配置是否成功。写完后我们可以用 cat 命令来查看 /data/htdocs/www/shop/mem.jsp 文件，文件内容如下所示：

```

<%
Runtime lRuntime = Runtime.getRuntime();
out.println("**** BEGIN MEMORY STATISTICS **** <br/>");
out.println("Free Memory: " + lRuntime.freeMemory() + "<br/>");
out.println("Max    Memory: " + lRuntime.maxMemory() + "<br/>");
out.println("Total Memory: " + lRuntime.totalMemory() + "<br/>");
out.println("Available Processors : " + lRuntime.availableProcessors() + "<br/>");

```

```
out.println("*** END MEMORY STATISTICS *** ");
%>
```

输入 `http://www.1paituan.com/mem.jsp`, 得到如下结果, 则证明 Nginx + Tomcat 设置成功。

```
*** BEGIN MEMORY STATISTICS ***
Free Memory: 54923016
Max Memory: 920911872
Total Memory: 76480512
Available Processors : 4
*** END MEMORY STATISTICS ***
```

2) LVS + Keepalived 的配置详细过程略过, 具体可参考前面的内容。以下为主 LVS 上的配置文件, 从 LVS 只需更改 MASTER 为 BACKUP、优先级低于 100 即可。

我们用 Vim 来编辑 `/etc/keepalived/keepalived.conf` 文件, 如下所示:

```
! Configuration File for keepalived
global_defs {
    notification_email {
        yuhongchun027@163.com
    }
    notification_email_from sns-lvs@gmail.com
    smtp_server 127.0.0.1
    router_id LVS_DEVEL
}
vrrp_instance VI_1 {
    state MASTER
    interface eth0
    virtual_router_id 51
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1111
    }
    virtual_ipaddress {
        203.93.236.148
    }
}
virtual_server 203.93.236.148 80 {
    delay_loop 6
    lb_algo wrr
    lb_kind DR
    persistence_timeout 60
    protocol TCP
    real_server 203.93.236.146 80 {
        weight 3
        TCP_CHECK {
            connect_timeout 10
            nb_get_retry 3
            delay_before_retry 3
```



```

        connect_port 80
    }
}
real_server 203.93.236.147 80 {
    weight 3
    TCP_CHECK {
        connect_timeout 10
        nb_get_retry 3
        delay_before_retry 3
        connect_port 80
    }
}
}

```

3) 由于用到了 LVS + Keepalived, 所有机器上均得关闭 iptables 和 SELinux, 尤其是 iptables, 一定记得不要打开, 不然会影响 LVS 转发效果的。在两台 Nginx + Tomcat 机器上执行 realserver.sh 脚本, 绑定网站的 VIP 地址-203.93.236.148, 并设定防 arp 抵制功能。脚本 realserver.sh 的代码如下:

```

#!/bin/bash
SNS_VIP=203.93.236.148
./etc/rc.d/init.d/functions
case "$1" in
start)
    ifconfig lo:0 $SNS_VIP netmask 255.255.255.255 broadcast $SNS_VIP
    /sbin/route add -host $SNS_VIP dev lo:0
    echo "1" >/proc/sys/net/ipv4/conf/lo/arp_ignore
    echo "2" >/proc/sys/net/ipv4/conf/lo/arp_announce
    echo "1" >/proc/sys/net/ipv4/conf/all/arp_ignore
    echo "2" >/proc/sys/net/ipv4/conf/all/arp_announce
    sysctl -p >/dev/null 2>&1
    echo "RealServer Start OK"
    ;;
stop)
    ifconfig lo:0 down
    route del $SNS_VIP >/dev/null 2>&1
    echo "0" >/proc/sys/net/ipv4/conf/lo/arp_ignore
    echo "0" >/proc/sys/net/ipv4/conf/lo/arp_announce
    echo "0" >/proc/sys/net/ipv4/conf/all/arp_ignore
    echo "0" >/proc/sys/net/ipv4/conf/all/arp_announce
    echo "RealServer Stopped"
    ;;
* )
    echo "Usage: $0 {start|stop}"
    exit 1
esac
exit 0

```

4) 记得将以下这些文件添加至 Centos5.5 的自启动文件里, 防止服务器调试时重启而忘记启动服务了。添加到/etc/rc.local 文件里的内容为如下:

```
ulimit -SHn 65535
/usr/local/sbin/realserver start
/usr/local/tomcat/bin/startup.sh
/usr/local/webserver/nginx/sbin/nginx
```

另外附上一些服务器上常用的 SHELL 脚本，方便大家进行线上维护和调试工作。

□ Nginx 日志截断脚本，方便日志收集，脚本内容如下：

```
#!/bin/bash
# This script run at 00:00
# The Nginx logs path
logs_path="/data/logs/"

mkdir -p ${logs_path}$(date -d "yesterday" +%Y)/$(date -d "yesterday" +%m)/

mv ${logs_path}www_tomcat.log ${logs_path}$(date -d "yesterday" +%Y)/$(date -d "yesterday" +%m)/www_tomcat_$(date -d "yesterday" +%Y%m%d).log
/usr/local/webserver/nginx/sbin/nginx -s reload
```

□ Nginx 服务器日志统计脚本，记得分析时去掉 LVS 负载均衡器的公网 IP，代码如下：

```
#!/bin/bash

if [ $# -eq 0 ]; then
    echo "Error: please specify logfile."
    exit 0
else
    LOG = $1
fi

if [ ! -f $1 ]; then
    echo "Sorry,sir,I can't find this apache log file,pls try again!"
    exit 0
fi

#####
echo "Most of the ip:"
echo "-----"
awk '{ print $1 }' $LOG | sort | uniq -c | sort -nr | head -10
echo
echo
#####
echo "Most of the time:"
echo "-----"
awk '{ print $4 }' $LOG | cut -c 14-18 | sort | uniq -c | sort -nr | head -10
echo
echo
#####
echo "Most of the page:"
echo "-----"
```

```

awk '{print $11}' $LOG | sed 's/^.*\(.cn*\)\.\/\1/g' | sort | uniq -c | sort -rn | head -10
echo
echo
#####
echo "Most of the time / Most of the ip:"
echo "-----"
awk '{ print $4 }' $LOG | cut -c 14-18 | sort -n | uniq -c | sort -nr | head -10 > timelog

for i in `awk '{ print $2 }' timelog`
do
    num=`grep $i timelog | awk '{ print $1 }'`
    echo " $i $num"
    ip=`grep $i $LOG | awk '{ print $1 }' | sort -n | uniq -c | sort -nr | head -10`
    echo " $ip"
    echo
done
rm -f timelog

```

❑ 防止 SSH 暴力破解脚本，现在将机器放在公网上还是很麻烦的，一天到晚都有人用 SSH 试密码。这个脚本功能等同于 SHELL 版的 DenyHosts，代码如下：

```

#!/bin/bash
cat /var/log/secure | awk '/Failed/{print $(NF-3)}' | sort | uniq -c | awk '{print $2"="$1;}' > /
root/black.txt
DEFINE="20"
for i in `cat /root/black.txt`
do
    IP=`echo $i | awk -F= '{print $1}'`
    NUM=`echo $i | awk -F= '{print $2}'`
    if [ $NUM -gt $DEFINE ];
    then
        grep $IP /etc/hosts.deny > /dev/null
        if [ $? -gt 0 ];
        then
            echo "sshd: $IP" >> /etc/hosts.deny
        fi
    fi
done

```

Tomcat 的连接数相关配置：

在 Tomcat 配置文件 server.xml 的配置中，和连接数相关的参数有：

- ❑ minProcessors，最小空闲连接线程数，用于提高系统处理性能，默认值为 10。
- ❑ maxProcessors，最大连接线程数，即并发处理的最大请求数，默认值为 75。
- ❑ acceptCount，允许的最大连接数，应大于等于 maxProcessors，默认值为 100。
- ❑ enableLookups，是否反查域名，取值为 true 或 false。为了提高处理能力，应设置为 false。
- ❑ connectionTimeout，网络连接超时，单位为毫秒。设置为 0 表示永不超时，这样设置是有隐患的。通常可设置为 30 000 毫秒。其中和最大连接数相关的参数为 maxProcessors 和 acceptCount。如果要加大并发连接数，应同时加大这两个参数。Web Server 允许的最大连接数还受制于操作系统的内核参数设置，通常 Windows 是 2000 个左右，Linux 是 1000 个左右。

- `maxThreads = "150"`，表示最多同时处理 150 个连接（即 Tomcat 的并发数），此值可以更改为 800 ~ 1000。
- `minSpareThreads = "25"`，表示即使没有人使用也开这么多空线程等待。
- `maxSpareThreads = "75"`，表示如果最多可以空 75 个线程，例如某时刻有 80 人访问，之后没有人访问了，则 Tomcat 不会保留 80 个空线程，而是关闭 5 个空的。
- `acceptCount = "100"`，当同时连接的人数达到 `maxThreads` 时，还可以接受排队的连接数，超过这个连接数则直接返回拒绝连接。

Tomcat 的其他相关参数，如下所示：

- `port`，Tomcat 服务器监听的端口号。
- `maxHttpHeaderSize`，Http 的 Header 的最大限制。
- `enableLookups = "false"`，使用允许 DNS 查询，通常情况下设置为 `false`。
- `redirectPort`，服务器在处理 http 请求时收到一个 SSL 传输请求后重定向的端口号。
- `compression = "on"`，打开压缩功能。
- `compressionMinSize`，启用压缩的输出内容大小，这里面默认为 2KB。
- `compressableMimeType`，压缩类型。

整个项目上线后还是比较稳定的，速度也非常快，由于目前还在内测阶段，并发数并不多。后期等用户数上去以后再根据具体情况跟 Java 开发人员一起进行 Tomcat 和 MySQL 数据库的优化工作。当然还是要注意防火墙安全和代码安全问题，这些都是后期维护工作的重点内容。

6.4.4 项目案例四：生产环境下的高可用 NFS 文件服务器

某日例行检查服务器时，发现我的 DRBD + Heartbeat + NFS 文件服务器已经稳定运行了 661 天（两台 Centos5.3 i386，DELL2950 机器，用于作图片及代码存放文件服务器），相当稳定和高效。鉴于 DRBD 已作为 MySQL 官方推荐的实现 MySQL 高可用的一种非常重要的方式，建议大家掌握这个知识点。如果没有线上环境的读者也不要着急，大家完全可以通过 VMware workstation6.0 + Centos5.x 系统来实现。在介绍过程中，如果有需要注意的地方我会重点说明，整个测试过程参考了线上文档。

分布式复制块设备（Distributed Replicated Block Device，DRBD）是一种基于 Linux 的软件组件，它是由内核模块和相关程序组成的，可通过网络镜像促进共享存储系统的替换。也就是说：当你将数据写入本地 DRBD 设备上的文件系统时，数据会同时被发送到网络中的另外一台主机之上，并以完全相同的形式记录在一个文件系统中（实际上文件系统的创建也是由 DRBD 的同步来实现的）。本地节点（主机）与远程节点（主机）的数据可以保证实时同步，并且可以保证 I/O 的一致性。所以当本地节点的主机出现故障时，远程节点的主机上还保留着一份完全相同的数据，可以继续使用，以达到高可用的目的。

DRBD 的工作原理如图 6-8 所示。

我们可以这样理解 DRBD，它其实就是网络 RAID-1，两台服务器中就算其中的某台因电源或主板损坏而宕机也不会对数据有任何影响（可以用硬盘 RAID-1 来理解 DRBD），而真正的热切换可以通过 Heartbeat 来实现，这样的切换过程跟 Keepalived 类似，非常短暂且不需要人为干预。

DRBD 需要构建在底层设备之上，而且还需要构建出一个块设备来。对于用户来说，一个

DRBD 设备，就像是一块物理的磁盘，可以在磁盘内创建文件系统。DRBD 所支持的底层设备如下：

- 一个磁盘，或者是磁盘的某一个分区。
- 一个 soft RAID 设备。
- 一个 LVM 的逻辑卷。
- 一个 EVMS (Enterprise Volume Management System, 企业卷管理系统) 的卷。

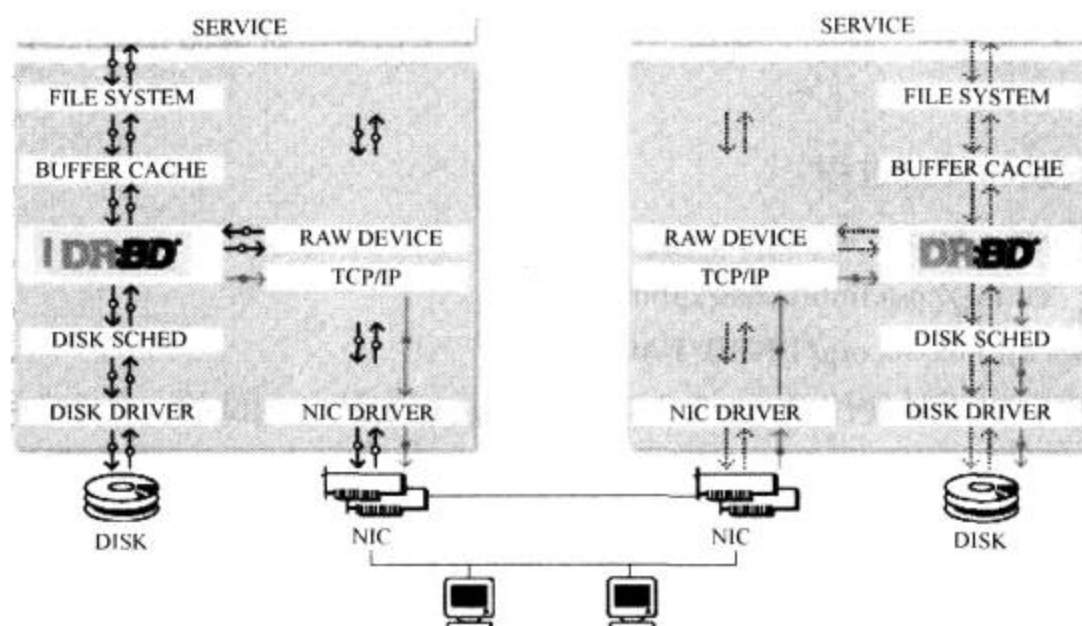


图 6-8 DRBD 工作原理图

- 其他任何的块设备。

我们的线上环境中采用的是第一种，即用单独的磁盘来做的 DRBD。

下面介绍 DRBD + Heartbeat + NFS 的详细配置过程。

1. 做好整个环境的准备工作

两台服务器均要做好准备工作，比如设置好 hosts 文件及进行 ntpd 对时。

primary 服务器：centos1.cn7788.com，单独拿一块硬盘 sdb 作为 DRBD。

secondary 服务器：centos2.cn7788.com，单独拿一块硬盘 sdb 作为 DRBD。

网络拓扑很简单，如下所示：

centos1.7788.com eth0: 192.168.1.103，物理 bridge 连接，eth1: 10.0.0.1（心跳线）

centos2.7788.com eth0: 192.168.1.104，物理 bridge 连接，eth1: 10.0.0.2（心跳线）

DRBD 对外的 VIP 地址是：192.168.1.105，这是通过 Heartbeat 来实现的，原理跟 Keepalived 类似，它通过在某台服务器的 eth0: 0 上绑定，如果遇到故障就转移，达到高 HA 的目的。这同时也是对外提供 NFS 服务的 IP。

关于/etc/hosts 的配置，在两台机器上是一样的，不需要太复杂，只须配置心跳部分即可，即：

```
192.168.1.103 centos1.cn7788.com centos1
192.168.1.104 centos2.cn7788.com centos2
```

本着简单高效的原则，两台 DELL 2850 服务器之间用交叉线连接。

DRBD 对服务器的性能要求不高，为了磁盘文件的安全，我都将其做成了 RAID5，操作系统为 Centos5.5 x86_64，在线上我们也用过 Centos5.4 i386，效果也很好。下面的演示过程中会以 Centos5.5 x86_64 版本为例来说明，建议采用我所推荐的版本，毕竟我们的 DRBD + Heartbeat 环境已经

稳定运行两年多了。

另外，关于时间同步问题，考虑到线上环境的严谨性，建议进行相关操作，命令如下：

```
ntpdate ntp.api.bz
```

最后，如果要想让 hosts 名字永久生效，建议重启机器，核对检查所有配置是否生效。最重要的心跳线一定要保持 ping 通状态，我们在工作中后期对 IP 进行了迁徙工作，发现只要心跳线不被人拔掉，基本上不会发生“脑裂”现象。另外，记得关掉两台机器的 iptables 和 SELinux，以免影响结果。

2. DRBD 的安装和配置过程

DRBD 的官方网站：<http://www.drbd.org/>

源码下载地址：<http://oss.linbit.com/drbd>

FAQ：<http://www.linux-ha.org/DRBD/FAQ>

1) 安装依赖库，以下是使用 Centos5.3 i386 系统进行安装与测试的，命令如下：

```
yum install gcc gcc-c++ make glibc flex devel devel-kenel
```

2) 在 primary 和 secondary 上都要使用相同的安装方法，如下所示：

```
cd /usr/local/src
wget http://oss.linbit.com/drbd/8.0/drbd-8.0.6.tar.gz
tar zxvf drbd-8.0.6.tar.gz
cd drbd-8.0.6
make DKDIR=/usr/src/kernel
make install
```

这里要说明一下，如果出现“Could not determine uts_release”错误。我们要检查是否安装新的内核，还要记得重启机器，不然也不能安装成功，命令如下所示：

```
make install
```

3) DRBD 程序安装完成后主要生成命令 drbdsetup、drbdadmin，配置文件/etc/drbd.conf、启动文件/etc/init.d/drbd 及模块文件：drbd.ko（在编译好的安装包目录下的 drbd 下可以找到）。我们可以用如下命令查看：

```
ll /lib/modules/2.6.18-238.9.1.el5/kernel/drivers/block/
total 3652
drwxr-xr-x 2 root root 4096 May 15 00:45 aoe
-rwxr--r-- 1 root root 141952 Apr 13 08:31 cciss.ko
-rwxr--r-- 1 root root 71712 Apr 13 08:31 cpqarray.ko
-rwxr--r-- 1 root root 39264 Apr 13 08:31 cryptoloop.ko
-rwxr--r-- 1 root root 128520 Apr 13 08:31 DAC960.ko
-rw-r--r-- 1 root root 2858045 May 15 00:54 drbd.ko
-rwxr--r-- 1 root root 142800 Apr 13 08:31 floppy.ko
-rwxr--r-- 1 root root 56824 Apr 13 08:31 loop.ko
-rwxr--r-- 1 root root 51984 Apr 13 08:31 nbd.ko
drwxr-xr-x 2 root root 4096 May 15 00:45 paride
-rwxr--r-- 1 root root 76216 Apr 13 08:31 pktcdvd.ko
-rwxr--r-- 1 root root 61744 Apr 13 08:31 sx8.ko
```

```
-rwxr--r-- 1 root root 45648 Apr 13 08:31 virtio_blk.ko
```

所有命令和配置文件都可以在源码包编译成功的目录下面找到。

4) DRBD 采用的是模块控制的方式, 所以先要加载 drbd.ko 模块, 命令如下:

```
modprobe drbd
```

查看 DRBD 模块是否已经加载到内核中了, 有的话表示已成功加载到内核中了, 命令如下:

```
lsmod | grep drbd
```

```
drbd                226608  0
```

5) 确认两台要镜像的机器是否正常, 它们之间的网络是否通畅, 需要加载的硬盘是否处于 umount 状态。

6) 在两台主机上都创建硬件设备 DRBD, 命令如下:

```
mknod /dev/drbd0 b 147 0
```

我们的两台主机都只有一块 DRBD 设备, 所以这里不需要再创建多个了。

7) 两台机器将 /dev/sdb1 互为镜像 (两台机器的配置相同), 这里只列出 centos1 的操作步骤, centos2 机器的操作步骤类似, 具体步骤如下所示:

```
#yum -y install portmap
```

```
#yum -y install nfs
```

```
#mkdir /drbd
```

创建共享目录, 我们用 Vim 可以编辑 /etc/exports 文件, 内容如下:

```
/drbd 192.168.1.0/255.255.255.0 (rw,no_root_squash,no_all_squash,sync)
```

只允许 192.168.1.0 网段的机器共享, 这个也是基于安全考虑的。

然后我们配置 NFS 涉及的 portmap 和 NFS 服务, 如下所示:

```
service portmap start
```

```
chkconfig portmap on
```

```
chkconfig nfs off
```

NFS 不需要启动, 也不需要设置成开机自动运行, 这些都将由后面的 Heartbeat 来完成。

8) 配置 DRBD。DRBD 运行时, 会读取一个配置文件 /etc/drbd.conf, 这个文件里描述了 DRBD 设备与硬盘分区的映射关系, 以及 DRBD 的一些配置参数。编辑 /etc/drbd.conf 文件, 内容如下:

```
resource r0 {
```

```
protocol C;
```

```
startup { wfc - timeout 0; degr - wfc - timeout 120; }
```

```
disk { on - io - error detach; }
```

```
net {
```

```
    timeout 60;
```

```
    connect - int 10;
```

```
    ping - int 10;
```

```
    max - buffers 2048;
```

```
    max - epoch - size 2048;
```

```
}
```

```

syncer { rate 30M; }

on centos1.cn7788.com {
    device    /dev/drbd0;
    disk      /dev/sdb;
    address   10.0.0.1:7788;
    meta-disk internal;
}
on centos2.cn7788.com {
    device    /dev/drbd0;
    disk      /dev/sdb;
    address   10.0.0.2:7788;
    meta-disk internal;
}
}

```

resource r0 表示创建的资源名字。

syncer C 表示采用 C 协议，如果收到远程主机的写入确认，则认为写入完成。syncer 选项是设备主备节点同步时的网络速率最大值。

每个主机的说明都以 on 开头，然后是各自的主机名，再后面的 {} 里是这个主机的配置，监听端口为 7788。

meta-disk internal 表示在同一个局域网内。

9) 启动 DRBD，激活前面配置的 DRBD 资源 r0（两个节点都要执行）。在启动 DRBD 之前，需要分别在两台主机的 hdb1 分区上创建供 DRBD 记录信息的数据块，分别在两台主机上执行如下命令：

```
drbdadm create-md r0
```

命令显示如下结果：

```

Valid meta-data already in place, recreate new?
[need to type 'yes' to confirm] yes
Creating meta data...
initialising activity log

NOT initialized bitmap (32 KB)

New drbd meta data block sucessfully created.

```

创建 r0 的资源，其中 r0 是我们在 drbd.conf 里定义的资源名称，最后一行显示创建 r0 资源成功。

现在可以启动 DRBD 了，分别在两台主机上执行，如下所示：

```
/etc/init.d/drbd start
```

设置 DRBD 开机时自动启动，命令如下：

```
chkconfig drbd on
```

系统会在 level2、leve3、level5 运行状态时自动启动 DRBD，如果是 level5（即图形状态下），此

条命令也是适用的。

现在可以查看 DRBD 当前的状态，在 centos1 上执行如下命令：

```
cat /proc/drbd
```

此命令执行后结果如下所示：

```
version: 8.0.0 (api:86/proto:86)
SVN Revision: 2713 build by root@ centos1,2008-06-27 14:07:14
1: cs:Connected st:Secondary/Secondary ds:Inconsistent/Inconsistent C r—
ns:0 nr:0 dw:0 dr:0 al:0 bm:0 lo:0 pe:0 ua:0 ap:0
resync: used:0/31 hits:0 misses:0 starving:0 dirty:0 changed:0 act_log: used:0/257 hits:0 mis-
ses:0 starving:0 dirty:0 changed:0
```

第一行的 st 表示两台主机的状态都是备机状态。ds 是磁盘状态，显示其状态“不一致”。这是因为 DRBD 无法判断哪一方为主机，应该以哪一方的磁盘数据作为标准数据。

所以我们需要初始化一个主机，那么需要在 centos1 上执行。

10) 初始化 centos1 (这步只要在主节点上操作)：

```
drbdsetup /dev/drbd0 primary -o
drbdadm primary r0
```

第一次设置主节点时用 drbdadm 命令会失败，所以先用 drbdsetup 来进行操作，以后就可以用 drbdadm 了。

再次查看 DRBD 当前的状态，命令如下所示：

```
cat /proc/drbd
version: 8.0.6 (api:86/proto:86)
SVN Revision: 2713 build by root@ centos1,2008-06-27 14:07:14
1: cs:SyncSource st:Primary/Secondary ds:UpToDate/Inconsistent C r—
ns:18528 nr:0 dw:0 dr:18528 al:0 bm:1 lo:0 pe:0 ua:0 ap:0
[ > ..... ] sync'ed: 0.3% (8170/8189)M
finish: 6:46:43 speed: 336 (324) K/sec
resync: used:0/31 hits:1156 misses:2 starving:0 dirty:0 changed:2
act_log: used:0/257 hits:0 misses:0 starving:0 dirty:0 changed:0
```

现在主备机状态分别是“主/备”，主机磁盘状态是“实时”，备机状态是“不一致”。

在第3行中可以看到数据正在同步，即主机正在将磁盘上的数据传递到备机上，现在的进度是0.3%。

设置完之后的第一次同步耗时会比较长，因为需要把整个分区的数据全部同步一遍。

第一次同步完成之后，就可以对 DRBD 的设备创建文件系统了。

稍等一段时间，在数据同步完成之后再查看一下两台机器的 DRBD 状态，命令分别如下：

```
[root@ centos1 ~]# service drbd status
SVN Revision: 3048 build by root@ centos1.cn7788.cn,2010-01-20 06:09:12
0: cs:Connected st:Primary/Secondary ds:UpToDate/UpToDate C r—

[root@ centos2 ~]# service drbd status
SVN Revision: 3048 build by root@ centos2.cn7788.cn,2010-01-20 06:09:02
0: cs:Connected st:Secondary/Primary ds:UpToDate/UpToDate C r—
```

现在磁盘状态都是“实时”，表示数据同步完成了。

在查看 DRBD 的实时状态时，我喜欢用以下语法（当然 `cat /proc/drbd` 也是可以的）：

```
service drbd status
```

11) DRBD 的使用。现在可以把主机上的 DRBD 设备挂载到一个目录上使用。备机的 DRBD 设备无法被挂载，因为它是用来接收主机数据的，由 DRBD 负责操作。

在 centos1 主服务器上执行如下命令：

```
mkfs.ext3 /dev/drbd0
mount /dev/drbd0 /drbd
```

现在，就可以对/drbd 分区进行读写操作了。

注意 secondary 节点上不允许对 DRBD 设备进行任何操作，包括只读。所有的读写操作只能在 primary 节点上进行，只有当 primary 节点挂掉时，secondary 节点才能提升成为 primary 节点，继续进行读写操作。

截至这步，DRBD 的安装就算成功了。那么我们究竟如何保证 DRBD 机器的高可用呢？这里就要用 Heartbeat 来实现了。另外，如果启动了 Heartbeat 服务，就不需要再手动 mount 了，Heartbeat 会自动 mount。

3. Heartbeat 的配置过程

1) 在两台主机上分别安装 Heartbeat，命令如下：

```
yum -y install heartbeat
```

奇怪的是，此命令要执行两次，不然 Heartbeat 安装不上去，这个也算是安装 Heartbeat 的 Bug 吧。

整个故障描述如下（很明显第一次安装 Heartbeat 包没有成功）：

```
error: %pre(heartbeat-2.1.3-3.el5.centos.x86_64) scriptlet failed,exit status 9
error: install: %pre scriptlet failed (2), skipping heartbeat-2.1.3-3.el5.centos
Installed:
heartbeat.x86_64 0:2.1.3-3.el5.centos Dependency Installed:
heartbeat-pils.x86_64 0:2.1.3-3.el5.centos heartbeat-stonith.x86_64 0:2.1.3-3.el5.centos
Complete!
```

本来要安装 4 个软件包，结果只安装成功了 3 个，所以必须要再执行一次如下命令：

```
yum -y install heartbeat
```

其中 Heartbeat 配置共涉及如下 4 个文件：

```
/etc/ha.d/ha.cf
/etc/ha.d/haresources
/etc/ha.d/authkeys
/etc/ha.d/resource.d/killnfsd
```

2) 两个节点的配置文件都是一样，/etc/ha.d/ha.cf 文件内容如下：

```
logfile          /var/log/ha-log
#定义 HA 的日志名字及存放位置
logfacility      local0
keepalive        2
#设定心跳(监测)时间为 2 秒
deadtime         5
#死亡时间定义为 5 秒
ucast            eth1 10.0.0.2
#采用单播方式,IP 地址指定为对方 IP
auto_failback    off
#服务器正常后由主服务器接管资源,另一台服务器放弃该资源
node             centos1.cn7788.com centos2.cn7788.com
#定义节点
```

3) 编辑双机互连验证文件 `authkeys`, 我们直接用 Vim 来编辑, 如下所示:

```
vim /etc/ha.d/authkeys
```

文件内容如下:

```
auth 1
1 crc
```

需要将 `/etc/ha.d/authkeys` 设为 600 的权限, 命令如下:

```
chmod 600 /etc/ha.d/authkeys
```

4) 编辑集群资源文件 `/etc/ha.d/haresources`, 内容如下所示:

```
centos1.cn7788.com IPaddr::192.168.1.108/24/eth0 drbdisk::r0 Filesystem::/dev/drbd0::/drbd::
ext3 killnfsd
```

此文件在两台机器上的配置是一样的, 强烈建议不要将另一台机器上配置成 `centos2.7788.com`, 最好的做法是先只在 `centos1` 机器上配置, 然后将文件 `scp` 或 `rsync` 配置到 `centos2` 机器上面。

5) 编辑脚本文件 `killnfsd`, 目的其实就是为了重启 NFS 服务。这是因为 NFS 服务切换后, 必须重新 `mount` 一下 NFS 共享出来的目录, 否则会出现 `stale NFS file handle` 的错误。我们直接用 Vim 来编辑 `/etc/ha.d/resource/killnfsd` 文件, 如下所示:

```
killall -9 nfsd; /etc/init.d/nfs restart; exit 0
```

给此文件 755 权限, 如下所示:

```
chmod 755 /etc/ha.d/resource.d/killnfsd
```

以上命令的作用是给予 `killnfsd` 执行的权限。

6) 在两个节点上启动 Heartbeat, 可以先在主节点上启动, 命令如下:

```
[root@ centos1 /]# service heartbeat start
[root@ centos2 /]# service heartbeat start
```

这时就可以在另外的机器上面正常挂载 `192.168.1.108:/drbd/` 到自己的 `/mnt/data` 下进行读写了, 客户端会认为这仅仅是一个提供 NFS 的机器。`192.168.1.108` 是 Heartbeat 产生的 VIP 地址, 也是对外提供 NFS 服务的地址。

4. 进行测试数据工作和故障模拟

虽然我的线上环境已经很稳定了，但为了让大家熟悉 DRBD + Heartbeat，还是按步骤进行了相关操作。建议完成如下测试后再看 Heartbeat 是否能做到真正的热切换。

1) 在另一台 FreeBSD8 下挂载 192.168.1.108: /drbd，在向里面写数据时，忽然重新启动主 DRBD，看此时写数据是否受到了影响。你可能会发现 DRBD + Heartbeat 正常切换还是需要些时间的。

2) 正常状态下将 Primary 关机，然后看数据有无问题，观察 DRBD 的 status；然后等主机启动后，再观察变化，然后再将 secondary 关机，再启动，观察 DRBD 变化及 Heartbeat 起了作用没有。

3) 假设此时把 Primary 的 eth0 给 ifdown 了，然后直接在 Secondary 上进行主 Primary 主机的提升，并且也给 mount 了，你可能会发现在 Primary 上测试拷入的文件确实同步过来了。之后把 primary 的 eth0 恢复后，看看有没有自动恢复主从关系。经过支持查询，你可能会发现 DRBD 检测出现了 Split-Brain 的状况，两个节点各自都 standalone 了。故障描述如下：Split-Brain detected, dropping connection! 这即是传说中的“脑裂”。DRBD 官方推荐手动恢复（生产环境下出现这个问题的几率很低，谁会故意去触动生产中的服务器网线呢？谁又会像这样去折腾线上的生产服务器呢？）。我们在工作中进行 IP 迁徙时，发现只要不触动心跳线，产生 Split-Brain 情况的几率真的很低。

以下为手动解决 Split-Brain 的方法。

a) 在 Secondary 主机上执行如下命令：

```
drbdadm secondary r0
drbdadm disconnect all
drbdadm ----discard-my-data connect r0
```

b) 在 Primary 主机上执行如下命令：

```
drbdadm disconnect all
drbdadm connect r0
```

4) 假设 Primary 主机因为硬件损坏，需要将 Secondary 提升成 Primary 主机，应该如何处理呢？方法如下：

a) 在 Primary 主机上，先要卸掉 DRBD 设备，命令如下：

```
umount /d
```

然后将主机降为备机，命令如下：

```
[root@ centos1 /]# drbdadm secondary r0
[root@ centos1 /]# cat /proc/drbd
1: cs:Connected st:Secondary/Secondary ds:UpToDate/UpToDate C r---
```

现在，两台主机都是备机了，我们可以通过 st 这项观察到此信息。

b) 将 centos2 备机升级为 Primary 主机，如下所示：

```
[root@ centos2 /]# drbdadm primary r0
[root@ centos2 /]# cat /proc/drbd
1: cs:Connected st:Primary/Secondary ds:UpToDate/UpToDate C r---
```

现在备机就成功转为主机了。由于 centos2 上面也有完整的数据，所以整个切换过程不会发生

数据丢失的现象，保证了数据的高可用性。

DRBD + Heartbeat + NFS 高可用文件服务器的整个构建过程还是比较复杂的，我这里有个建议：我们可以在局域网或 VMware Server 环境下先将整个过程测试几十次，这样大家自然而然就会清楚其中的逻辑顺序了。大家可以也像我一样将这些操作步骤形成工作文档，这样，在线上环境中实施时会非常容易，一步一步照做就行了。DRBD 是现在很流行的技术，很多公司极有可能会用它作为高可用文件服务器。不仅如此，DRBD + Heartbeat 也适合做生产环境下的高可用 MySQL 服务（这个也是 MySQL 官方推荐），所以我推荐大家掌握 DRBD + Heartbeat + NFS 的配置方法。

6.4.5 项目案例五：HAProxy 双机高可用方案之 HAProxy + Heartbeat

HAProxy + Heartbeat 配置过程如下。

1. 做好整个环境的准备工作

双机的硬件配置和文件配置跟我们前面配置 DRBD + Heartbeat + NFS 类似。

两台服务器 DELL 2950 均要做好准备工作，比如设置好 hosts 文件及进行 ntpd 对时。

网络拓扑很简单，如下所示：

ha1.cn7789.com eth0: 192.168.1.5, 物理 bridge 连接

ha2.cn7789.com eth0: 192.168.1.6, 物理 bridge 连接

DRBD 对外的 VIP 地址是：192.168.1.8，这是通过 Heartbeat 来实现的，原理跟 Keepalived 类似，它通过在某台服务器的 eth0: 0 上绑定，从而实现遇到故障就转移的功能，这样就达到高 HA 的目的。这同时也是对外提供 NFS 服务的 IP。

关于/etc/hosts 的配置，在两台机器上是一样的，不需要太复杂，即：

```
192.168.1.5 ha1.cn7789.com ha1
192.168.1.6 ha2.cn7789.com ha2
```

后面再跟上两台 Web 服务器，我们这里配置的 IP 分别是 192.168.1.11 和 192.168.1.12，系统为 FreeBSD 或 Centos 均可，Nginx 作为 Web 的配置过程这里就不说了。

2. HAProxy 和 Heartbeat 的安装过程

关于此安装过程，请大家参考前面的内容，这里就不重复了，我们主要是注意关键位置的改动。

1) 要建立 HAProxy 启动、重启、关闭等状态脚本，大家应该比较清楚在 Heartbeat 里，它的资源要支持这种脚本，我们的 HAProxy 脚本为/root/haproxy，我们给它执行权限，脚本内容如下所示：

```
#!/bin/sh
# chkconfig 35 on
# description: HAProxy is a TCP/HTTP reverse proxy which is particularly suited for high availability environments.

# Source function library.
if [ -f /etc/init.d/functions ]; then
    ./etc/init.d/functions
elif [ -f /etc/rc.d/init.d/functions ]; then
    ./etc/rc.d/init.d/functions
```

```

else
    exit 0
fi

# Source networking configuration.
./etc/sysconfig/network

# Check that networking is up.
[ ${NETWORKING} = "no" ] && exit 0

[ -f /usr/local/haproxy/conf/haproxy.cfg ] || exit 1

RETVAL=0

start() {
    /usr/local/haproxy/sbin/haproxy -c -q -f /usr/local/haproxy/conf/haproxy.cfg
    if [ $? -ne 0 ]; then
        echo "Errors found in configuration file."
        return 1
    fi

    echo -n "Starting HAproxy: "
    daemon /usr/local/haproxy/sbin/haproxy -D -f /usr/local/haproxy/conf/haproxy.cfg -p /var/
run/haproxy.pid
    RETVAL = $?
    echo
    [ $RETVAL -eq 0 ] && touch /var/lock/subsys/haproxy
    return $RETVAL
}

stop() {
    echo -n "Shutting down HAproxy: "
    killproc haproxy -USR1
    RETVAL = $?
    echo
    [ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/haproxy
    [ $RETVAL -eq 0 ] && rm -f /var/run/haproxy.pid
    return $RETVAL
}

restart() {
    /usr/local/haproxy/sbin/haproxy -c -q -f /usr/local/haproxy/conf/haproxy.cfg
    if [ $? -ne 0 ]; then
        echo "Errors found in configuration file, check it with 'haproxy check'."
        return 1
    fi
    stop
    start
}

check() {

```

```

    /usr/local/haproxy/sbin/haproxy -c -q -V -f /usr/local/haproxy/conf/haproxy.cfg
}
rhstatus() {
    status haproxy
}

condrestart() {
    [ -e /var/lock/subsys/haproxy ] && restart || :
}

# See how we were called.
case "$1" in
    start)
        start
        ;;
    stop)
        stop
        ;;
    restart)
        restart
        ;;
    reload)
        restart
        ;;
    condrestart)
        condrestart
        ;;
    status)
        rhstatus
        ;;
    check)
        check
        ;;
    * )
        echo $"Usage: haproxy {start|stop|restart|reload|condrestart|status|check}"
        RETVAL=1
esac

exit $RETVAL

```

我们将此脚本文件分别拷贝至/etc/init.d/和/etc/ha.d/resource.d下，将 HAProxy 配置成服务启动方式。

2) /usr/local/haproxy/conf/haproxy.cfg 文件的内容如下所示（两台 HAProxy 机器的配置内容一样）：

```

global
    log 127.0.0.1    local0
    maxconn 4096
    chroot /usr/local/haproxy
    uid 99

```

```

gid 99
daemon
nbproc 1
pidfile /usr/local/haproxy/logs/haproxy.pid
defaults
    log      127.0.0.1      local3
    mode     http
    option httplog
    option httpclose
    option dontlognull
    option forwardfor
    option redispatch
    retries 2
    maxconn 2000
    balance source
    stats uri /haproxy-stats
    timeout 5000
    clitimeout 50000
    srvtimeout 50000
listen web_proxy 192.168.1.8:80
    option httpchk HEAD /index.php HTTP/1.0
    server web1 192.168.1.11:80 cookie applinst1 check inter 2000 rise 2 fall 5
    server web2 192.168.1.12:80 cookie applinst2 check inter 2000 rise 2 fall 5

```

HAProxy 加上日志支持，设置过程如下：

我们用 vim 编辑/etc/syslog.conf 文件，增加内容如下：

```

local3.*      /var/log/haproxy.log
local0.*      /var/log/haproxy.log

```

继续用 vim 编辑/etc/sysconfig/syslog 文件，修改内容如下：

```
SYSLOGD_OPTIONS = " -r -m 0"
```

重启 syslog 服务，命令如下：

```
service syslog restart
```

3) Heartbeat 的主配置文件/etc/ha.d/ha.cf 文件我们按照如下内容配置。

在 ha1.cn7789.com 上，我们可以用 cat /ec/ha.d/ha.cf 查看内容，详细参数配置这里就不解释了，大家可以参考前面的内容，如下所示：

```

logfile /var/log/ha-log
logfacility local0
keepalive 2
deadtime 5

ucast eth0 192.168.1.6
auto_failback off
node ha1.cn7789.com
node ha2.cn7789.com

```


ha2.cn7789.com 的 ha.cf 文件内容如下：

```
logfile /var/log/ha - log
logfacility local0
keepalive 2
deadtime 5
ucast eth0 192.168.1.5
auto_failback off
node hal.cn7789.com
node ha2.cn7789.com
```

4) 两台机器的密码验证文件/etc/ha.d/authkeys 内容是一致的（记得分配 600 权限），如下所示：

```
auth 1
1 crc
```

5) 两台机器的 haresources 文件配置一样，内容如下：

```
hal.cn7789.com 192.168.1.8 haproxy
```

3. 分别将 HAProxy 服务和 Heartbeat 服务设成自启动方式

将 HAProxy 服务和 Heartbeat 服务设成自启动方式的命令如下：

```
service haproxy start
chkconfig haproxy on
service heartbeat start
chkconfig heartbeat on
```

4. 验证此架构

在客户机正常访问网站地址（即 <http://192.168.1.8>）的过程中，我们将主 HAProxy 机器（即 hal.cn7789.com 机器）拔掉网线和重启服务器，或者干脆关机，看是否影响客户机访问后端的 Web 机器。以上步骤我自己测试过多次，证明 HAProxy + Heartbeat 双机方案确实是有效的。

关于 HAProxy + Heartbeat 这种负载均衡高可用架构，有些情况我也跟大家说明一下：

- HAProxy + Heartbeat 是客户要求的双机方案，由于他们以前也有别的应用服务器在用 HAProxy，所以也要求我们为其设计这样的高可用方案。
- 目前此套方案只是处于测试阶段，如果对 Linux 集群稳定性有要求的朋友我建议参考前面的 LVS + Keepalived 和 Nginx + Keepalived 项目方案。
- 为了保证 Heartbeat 不出现“脑裂”的问题，请确保机房的心跳线无人触动，这点其实容易实现，现在机房都应该有比较严格的管理制度。其实在实施过程中我也考虑过不用心跳线，心跳监控采用业务线路（即单网卡方式），但又担心后期网站流量过大对心跳线监控这块产生影响，从而发生“脑裂”的情况，所以最后决定还是采用服务器双网卡的方式。
- 如果大家想要将此架构用于生产环境，只有多进行测试，实践出真知，争取在实验阶段把该出的问题都找到，这样到了生产环境我们就能用最短的时间解决掉问题了。

- 在 Heartbeat 配置文件中我设置了 `auto_failback off` 参数，此参数的作用是让 Heartbeat 不发生抢占 VIP 地址的现象，即：主 HAProxy 重启或 Crash 后让出了 VIP 地址，此 VIP 地址被从 HAProxy 接管后就不会再让出来了。事实上，在生产环境中如果发生了这种现象，我们应该立即查找原因，看到底是服务器硬件引起的还是程序引起的，以免影响网站的正常运行。
- 此架构目前只为百万级 PV 流量的网站设计，如果有更高流量的需求，考虑到以后的扩展问题，我推荐大家采用 LVS + Keepalived → HAProxy 负载均衡器（多台）→ Nginx Web 集群的方案，这种方案参考了章文嵩先生在淘宝网上的海量图片存储系统，是基于线上环境的成熟架构，我们的网站应用这种架构是绝对没有问题的。

6.5 项目实践中 Linux 集群的总结和思考

下面总结了一些项目实践中关于 Linux 集群的几个重要知识点，希望能对大家的工作有所帮助。

1) 目前网站架构一般分为负载均衡层、Web 层和数据库层，其实一般我还会多加一层，即文件服务器层，因为随着网站的流量越来越多，文件服务器的压力也越来越大，DRBD + Heartbeat + NFS 架构在后期可能会觉得压力太大了，这时候我们可以考虑图片单独分离，或者在前端采用 Squid 服务器缓存集群的方法来缓解海量图片的压力。网站最前端的负载均衡层称之为 Director，它起分摊请求的作用，最常见的就是轮询，这个基本功能是每一款负载均衡器都自带的。

2) F5 是通过硬件的方式来实现负载均衡的，在 CDN 系统上用得较多，用于 Squid 反向加速集群的负载均衡，是专业的硬件负载均衡设备，尤其适合每秒新建连接数和并发连接数要求高的场景。LVS 和 HAProxy 是通过软件的方式来实现的，但其稳定性也相当强悍，在处理高并发的情况时有着相当不俗的表现。至于 Nginx，我现在较倾向于将其放在整个系统或网站的中间，让其作为中间层的负载均衡器，这样效果更好。

3) Nginx 对网络的依赖较小，理论上只要 ping 得通，网页访问正常，Nginx 就能连得通。Nginx 还能区分内外网，如果是同时拥有内外网的节点，就相当于单机拥有了备份线路。LVS 则比较依赖于网络环境，它所涉及项目的所有服务器都必须在同一机房的同一交换机上，我们有时不得不考虑单交换带来的单点故障问题。

4) 目前较成熟的负载均衡高可用技术有 LVS + Keepalived、Nginx + Keepalived，以前 Nginx 没有成熟的双机备份方案，但通过 SHELL 脚本监控是可以实现的。另外，如果考虑 Nginx 的双主负载均衡高可用，也可以通过 DNS 轮询的方式来实现，但由于我们目前的商务网站要考虑 Google 收录及域名备案还有其他的因素，所以暂时只能用单域名，目前只能采用 Nginx + Keepalived 方案了。Nginx + Keepalived 在我们的机房中已经稳定运行两年了，所以我向大家推荐这个方案，有相关工作需求的朋友可以放心实施。

5) 集群是指负载均衡后面的 Web 集群或 Tomcat 集群等，但有时候大家所谈论的 Linux 集群却泛指了整个系统架构，既包括了负载均衡器，也包括了后端的应用服务器集群等。其实我们可以这样来加以区分，如果专指小范围的集群概念，可以指 Web 集群、Squid 集群等，如果指广泛意义上的集群，我们可以接受 Linux 集群这种叫法，这样大家就不至于混淆了。

6) 负载均衡高可用中的高可用指的是实现负载均衡器的 HA，即一台负载均衡器坏掉后另一台可以在 <1 秒的时间内切换，最常用的软件就是 Keepalived 和 Heartbeat，在成熟的生产环境下负载均衡器方案有 LVS + Keepalived、Nginx + Keepalived。如果能保证 Heartbeat 的心跳线稳定的话，

Heartbeat + DRBD 也可应用在很成熟的生产环境下, 适合 NFS 文件服务器或 MySQL 的高可用环境。

7) LVS 的优势非常多, 比如: 抗负载能力强, 工作稳定性高 (因为有成熟的 HA 方案), 无流量的转发, 效率高, 基本上能支持所有的 TCP 应用 (当然包括 Web) 等。基于以上的优点, LVS 拥有不少的粉丝, 但它也有缺点, LVS 对网络的依赖性太大了, 在网络环境相对复杂的应用场景中, 我不得不放弃它而选用 Nginx。

8) Nginx 对网络的依赖性小, 而且它的正则表达式强大且灵活 (个人觉得比 HAProxy 的确实简单些), 其稳定的特点吸引了不少人, 而且配置起来也是相当的方便和简约, 在中小型项目的实施过程中我基本上都会考虑用它。当然, 如果资金充足, F5 是不二的选择。

9) 在大型网站架构中其实可以结合使用 F5/LVS 或 Nginx, 根据情况选择其中的两种或全部选择。如果因为预算的原因不选择 F5, 那么网站最前端的指向应该是 LVS, 也就是 DNS 的指向应为 LVS 均衡器, LVS 的优点令它非常适合这个任务。那些重要的 IP 地址, 最好交由 LVS 托管, 比如数据库的 IP、提供 Web 服务的 IP 等, 这些 IP 地址随着时间的推移, 使用面会越来越大。如果更换 IP 则故障会接踵而至, 所以将这些重要 IP 交给 LSV 托管是最为稳妥的。

10) VIP 地址是 Keepalived 或 Heartbeat 虚拟的一个 IP, 它是一个对外的公开 IP, 也是 DNS 指向的 IP, 所以在设计网站架构时, 你必须向你的 IDC 多申请一个对外 IP。如果是做 LVS + Keepalived 的纯公网架构, 那就最好购买一个 IP 网段吧。

11) 在实际项目实施过程中, 我发现 LVS 和 Nginx 对 HTTPs 的支持都非常好, 尤其是 LVS, 相对而言处理起来更为简便。

12) 如果是基于 LVS + Keepalived 及 Nginx + Keepalived 架构的 Web 网站发生故障时, 我们处理起来是很方便的。如果发生了系统故障或服务器相关故障, 可以将 DNS 指向后端的某台真实 Web 机器上, 达到短期处理故障的目的。对于广告网站和电子商务网站来说, 流量就是金钱, 这也是要将负载均衡高可用设计于此的原因。如果是大型的广告网站, 我建议最好直接上 CDN 系统。

13) 现在 Linux 集群被大家神化了, 其实它并不是很复杂, 关键看你的应用场景, 哪种适合就选用哪种。Nginx、HAProxy、LVS 和 F5 都不是神话, 都有自己本身的缺陷, 我们应该扬长避短, 最大限度地发挥它们的优势。

14) 另外关于 Session 共享的问题 (这也是一个老生常谈的问题了), Nginx 可以用 ip_hash 机制解决之, HAProxy 有 balance source, 而 F5/LVS 则可以用会话保持机制来解决这个问题。此外, 还可以将 Session 写进数据库, 这也是一个解决 Session 共享的好办法, 当然这也会加重数据库的负担, 这就要看系统架构师的取舍了。

15) 现在很多朋友都用 Nginx (尤其是作 Web 服务器), 其实在服务器性能优异且内存足够的情况下, Apache 的抗并发能力并不弱 (16GB 的内存下 Apache 过 2000 并发问题也不大, 当然配置文件也要优化), 整个网站的瓶颈应该还是在数据库方面。我建议大家全面了解 Apache 和 Nginx, 前端用 Nginx 作负载均衡器, 后端用 Apache 作 Web 应用服务器, 这样升级以前的单 Apache 网站时就可以实现平稳过渡, 给网站带来的影响也是最小的。

16) Heartbeat 的“脑裂”问题没有想象中的那么严重, 在线上环境下可以考虑使用它。DRBD + Heartbeat 算是成熟的应用了, 我建议掌握这个知识热点。我在相当多的场合中用此组合来替代 EMC 共享存储, 毕竟 30 多万元的价格并不是每个客户都能接受的。

17) 无论设计的方案有多么成熟, 还是建议配置 Nagios 监控机来实时监控服务器的情况。邮件和

短信报警都可以开启，毕竟手机可以随身携带。有条件的话还可以购买专门的商业扫描网站服务，例如 Alertbot，它会以 1 秒为频率扫描你的网站，如果发现没有 alive，它会向你的邮箱中发警告邮件。

18) 关于网站的安全性问题，我建议用硬件防火墙，比较推荐的是华赛或 Juniper 系列的防火墙，DDoS 的安全防护一定要到位（国内的 DDoS 攻击还是挺多的）。Linux 服务器本身的 iptables 和 SELinux 均可关闭。另外，端口开放得越少越好。

19) 测试网站的响应时间时我们可以用 <http://tools.pingdom.com>，我发现用了 LVS + Keepalived、Nginx + Keepalived 后并不影响网站的访问速度，各位不用多虑。Nginx 现在做反向加速也日趋成熟了，大家可尝试一下在自己的网站上用它来作反向加速服务器。

6.6 网站架构应关注和研究的方向

1) 我们的网站放在 IDC 机房内，首先考虑的就应该是如何防止 DDoS 攻击。DDoS 攻击虽然没什么技术含量，但真正攻击过来还是很让人烦躁的。在搭建网站或系统时，我们应该尽可能地了解和熟悉各种防火墙的技术指标参数，了解哪种防火墙防护 DDoS 的效果更好。为客户提供性价比最好的防火墙方案也是保证整套系统或网站成功的因素之一。

2) 业务逻辑设计要合理，尤其是程序代码层的相关设计，如果程序应用架构和业务实现不够优化，一个本来很简单的实现却绕了很多弯才实现，那么多强的硬件也没有用。

3) 也许是受张宴先生的影响，现在越来越多的朋友把注意力放在 Nginx 上了。前面已经说过，Apache 的抗并发能力并不弱。在生产环境下，如果我们的网站不是广告型网站、门户型网站或游戏型网站，2000 并发已经是一个很惊人的数字。另外这个仅仅是一台 Apache 的并发数，一个中等规模的网站，后端至少会有 3~4 台 Apache 的 Web 应用服务器，所以，全部加起来我们的网站差不多可以顶住上万的并发了，上万的并发量对网站根本没有什么大的影响。当然，如果换用 Nginx 作 Web 应用服务器更没有问题了。另外，就算并发量非常大，我们最前端的 F5/LVS 还是顶得住的，无非是在后端多加几台 Web 应用服务器。所以说，并发量过大不可能成为网站的瓶颈。

4) DRBD + Heartbeat + NFS 文件服务器在初期没什么压力，但随着网站的用户数和流量越来越大，它可能会感觉有些顶不住了，特别是用户频繁访问图片文件时。我们在公司内部也测试过 Google 的分布式文件系统，但一直没敢用于生产环境中，最后还是决定采用 Nginx 作中层代理，增加 Squid 反向代理服务器集群的方法来解决文件服务器的压力问题。另外，我还曾尝试采用域名分散策略的方法，例如使用 pics.xxx.com/images.xxx.com 来区分标记为 pics 或 images 的一系列文件，这些文件存储的时候依然按照标记，存到 pics 或 images 的服务器上。这个策略将区分机器的任务交由 DNS 服务器来执行，扩容时会相对轻松。应在 Web 项目初期就规划好这些，后期才转用域名策略的成本比较高甚至不可实现，大家可以注意下。其实如果网站是专业的图片服务器时文件服务器层的压力还是很大的，我们需要在这个上面投入足够多的硬件资源。

5) 将 Nginx 作为中层代理使用是一件性价比非常高的事情，在这一层面上 Nginx 负载均衡器基本上无对手，如果担心单点 Nginx 故障，我们可以设置 3 台以上的 Nginx 负载均衡器，而它们的 load balance 可以让 F5 或 LVS 来做。Nginx 在这层可以利用其强大的正则处理能力很完美地处理客户端对静态文件的访问，比如将 html、jpg、png、CSS 等交给后端的 Squid 集群处理，动态的 PHP 或 JSP 访问请求交给后端的 Tomcat 集群服务器处理，动静分离，最大化地发挥 Nginx 作为负载均衡器/反向代理的优势。Nginx 作为中层负载的拓扑图如图 6-9 所示。

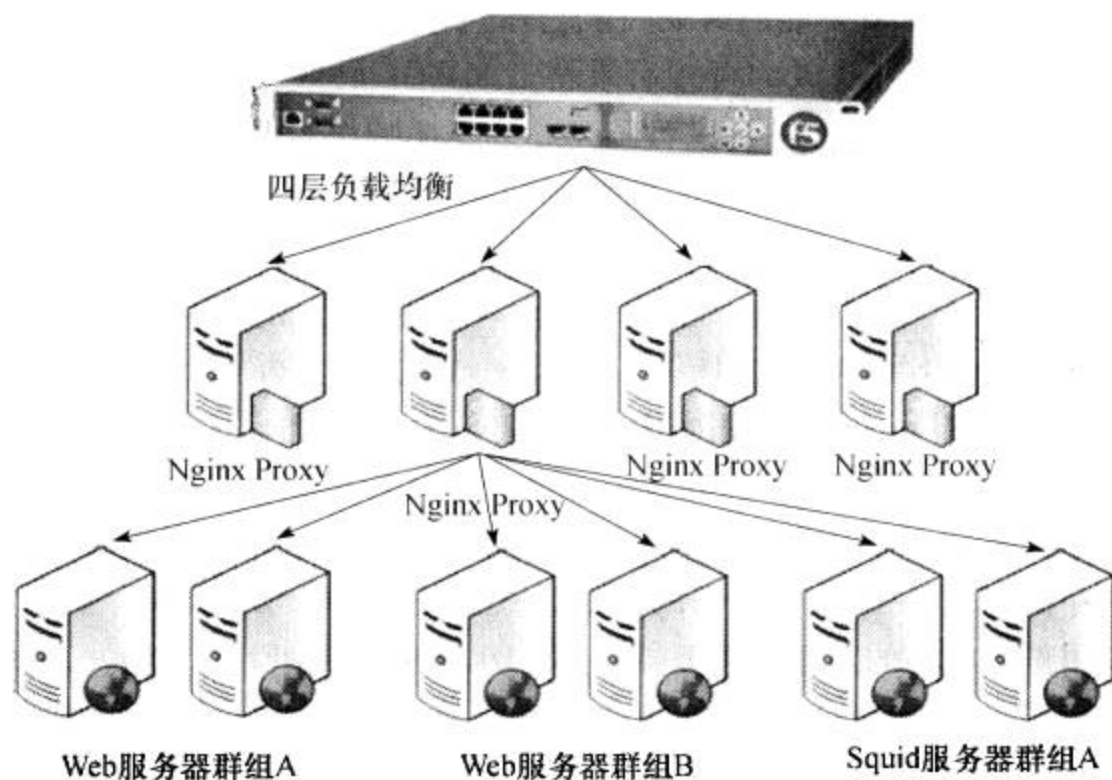


图 6-9 Nginx 作为中层负载均衡拓扑图

如果没有硬件的 F5 Big-IP 设备，我们也可以用软件 LVS 来实现，这样成本会相当低。Nginx 则利用其强大的正则功能，并根据 URL 或客户请求文件的后缀名来做动静分离，或者轮询不同的 Squid 反向代理群组。

6) 上线的项目在后期无论怎么优化或架构，最后其压力最大的肯定是 MySQL 数据库，尤其是那些动态网站。我们在维护时也发现，MySQL 数据库在频繁地读写，如何优化 MySQL 数据库一直是我们的关注和研究的方向，我也希望大家在工作中注意这个问题。

7) 系统或网站的构建、运维和调试并不只是一个人的事情，它是整个团队合作努力的结果，需要整个团队的开发人员、系统工程师和 DBA 及测试人员共同努力，要写出安全、效率高、优美的代码，需要花费开发人员大量的心血。至于 MySQL 数据库的调优，更需要长期的监测和大量的调试，这也是我们在工作中应该注意的问题。

6.7 MySQL数据库的优化

我们究竟应该如何对 MySQL 数据库进行优化呢？下面我就从 MySQL 对硬件的选择、MySQL 的安装、my.cnf 的优化、MySQL 如何进行架构设计及数据切分等方面来说明这个问题。需要说明一下，由于本人不是专业的 MySQL DBA，MySQL 架构设计这部分内容参考了简朝阳先生的著作《MySQL 性能调优与架构设计》。

6.7.1 服务器物理硬件的优化

在挑选硬件 MySQL 服务器时，我们应该从下面几个方面着重对 MySQL 服务器的硬件配置进行优化，也就是说将项目中的资金着重投入到如下几处：

1) 磁盘寻道能力（磁盘 I/O），我们现在用的都是 SAS15000 转的硬盘，用 6 块这样的硬盘作 RAID1 + 0。MySQL 每一秒钟都在进行大量、复杂的查询操作，对磁盘的读写量可想而知，所以，

通常认为磁盘 I/O 是制约 MySQL 性能的最大因素之一。对于日均访问量在 100 万 PV 以上的 Discuz! 论坛，如果磁盘 I/O 性能不好，造成的直接后果就是 MySQL 的性能会非常低下！解决这一制约因素可以考虑解决方案是：使用 RAID1+0 磁盘阵列，注意不要尝试使用 RAID-5，MySQL 在 RAID-5 磁盘阵列上的效率不会像你期待的那样快，如果资金允许，可以选择固态硬盘 SSD 来代替 SAS 硬件作 RAID1+0。

2) CPU 对于 MySQL 的影响也是不容忽视的，建议选择运算能力强悍的 CPU。推荐使用 DELL PowerEdge R710, Intel Xeon E5504（双四核），商家的卖点也是其强大的虚拟化和数据库能力。我现在比较喜欢用其做 Linux/FreeBSD 下的虚拟化应用，效果不错。

3) 对于一台使用 MySQL 的 Database Server 来说，建议服务器的内存不要小于 2GB，推荐使用 4GB 以上的物理内存。不过内存对于现在的服务器而言可以说是一个可以忽略的问题，如果是高端服务器，基本上内存都超过了 32GB，我们的数据库服务器都是 32GB DDR3。

我们在工作中用得比较多的数据库服务器是 HP DL580G5 和 DELL R710，其稳定性和性能都不错，特别是 DELL R710，我发现许多同行都是采用它作数据库的服务器的，所以在这里也向大家推荐一下。

6.7.2 MySQL 应该采用编译安装的方法

关于 MySQL 数据库的线上环境安装，我建议采取编译安装的方法，这样性能会有较大的提升。服务器系统则建议用 Centos5.5 x86_64，源码包的编译参数会默认以 Debug 模式生成二进制代码，而 Debug 模式给 MySQL 带来的性能损失是比较大的，所以当我们编译准备安装的产品代码时，一定不要忘记使用 `--without-debug` 参数禁用 Debug 模式。如果把 `--with-mysqld-ldflags` 和 `--with-client-ldflags` 两个编译参数设置为 `--all-static` 的话，可以告诉编译器以静态的方式编译，编译结果将得到最高的性能。使用静态编译和使用动态编译的代码相比，性能差距可能会达到 5% 至 10% 之多。在后面的章节我会跟大家分享我们线上 MySQL 数据库的编译参数，大家可以参考一下，然后根据自己的线上环境自行修改内容。

6.7.3 MySQL 配置文件的优化

如果解决了上述的服务器硬件制约因素，那么再来看看 MySQL 自身的优化是如何操作的。对 MySQL 自身的优化主要是对其配置文件 `my.cnf` 中的各项参数进行优化调整。下面我们介绍一些对性能影响较大的参数。

下面，我们根据以上推荐的硬件配置并结合一份已经优化好的 `my.cnf` 进行说明。

以下只列出 `my.cnf` 文件中 `[mysqld]` 段落里的内容，其他段落的内容对 MySQL 的运行性能影响甚微，比如 MySQL 的语言和日志配置选项，这里姑且忽略。

```
[mysqld]
```

`[mysqld]` 组中包括了 `mysqld` 服务启动的参数，它涉及的方面很多，其中有 MySQL 的目录和文件，通信、网络、信息安全，内存管理、优化、查询缓存区，还有 MySQL 日志设置等。

```
port = 3306
```

`mysqld` 服务运行时的端口号。

```
socket = /tmp/mysql.sock
```

socket 文件是在 Linux/Unix 环境下特有的，用户在 Linux/Unix 环境下客户端连接可以不通过 TCP/IP 网络而直接使用 unix socket 连接 MySQL。

```
skip-locking
```

避免 MySQL 的外部锁定，减少出错几率，增强稳定性。

```
skip-name-resolve
```

禁止 MySQL 对外部连接进行 DNS 解析，使用这一选项可以消除 MySQL 进行 DNS 解析的时间。但需要注意的是，如果开启该选项，则所有远程主机连接授权都要使用 IP 地址方式了，否则 MySQL 将无法正确处理连接请求！

```
back_log = 384
```

back_log 参数的值指出在 MySQL 暂时停止响应新请求之前，短时间内的多少个请求可以被存在堆栈中。如果系统在短时间内有很多连接，则需要增大该参数的值，该参数值指定到来的 TCP/IP 连接的监听队列的大小。不同的操作系统在这个队列的大小上有自己的限制。如果试图将 back_log 设定得高于操作系统的限制将是无效的。其默认值为 50。对于 Linux 系统而言，推荐设置为小于 512 的整数。

```
key_buffer_size = 384M
```

key_buffer_size 指定用于索引的缓冲区大小，增加它可得到更好的索引处理性能。对于内存在 4GB 左右的服务器来说，该参数可设置为 256MB 或 384MB。

注意 如果该参数值设置得过大反而会使服务器的整体效率降低！

```
max_allowed_packet = 4M
```

设定在网络传输中一次消息传输量的最大值。系统默认值为 1MB，最大值是 1GB，必须设定为 1024 的倍数，单位为字节。

```
thread_stack = 256K
```

设置 MySQL 每个线程的堆栈大小，默认值足够大，可满足普通操作。可设置范围为 128KB 至 4GB，默认为 192KB。

```
table_cache = 614K
```

table_cache 指示表高速缓冲区的大小。当 MySQL 访问一个表时，如果在 MySQL 表缓冲区中还有空间，那么这个表就被打开并放入表缓冲区，这样做的好处是可以更快速地访问表中的内容。一般来说，可以查看数据库运行峰值时间的状态值 Open_tables 和 Opened_tables，用以判断是否需要增加 table_cache 的值，即如果 Open_tables 接近 table_cache 的时候，并且 Opened_tables 这个值在逐步增加，那就要考虑增加这个值的大小了。

```
sort_buffer_size = 6M
```

设定查询排序时所能使用的缓冲区大小，系统默认大小为 2MB。从 5.1.23 版本开始，在除了

Windows 之外的 64 位平台上可以超出 4GB 的限制。

注意 该参数对应的分配内存是每个连接独占的，如果有 100 个连接，那么实际分配的总排序缓冲区大小为 $100 \times 6 = 600\text{MB}$ 。所以，对于内存在 4GB 左右的服务器来说，推荐将其设置为 6MB ~ 8MB。

```
read_buffer_size = 4M
```

读查询操作所能使用的缓冲区大小。和 `sort_buffer_size` 一样，该参数对应的分配内存也是每个连接独享。

```
join_buffer_size = 8M
```

联合查询操作所能使用的缓冲区大小，和 `sort_buffer_size` 一样，该参数对应的分配内存也是每个连接独享。

```
mysam_sort_buffer_size = 64M
```

设置在 REPAIR TABLE 或用 CREATE INDEX 创建索引或 ALTER TABLE 的过程中排序索引所分配的缓冲区大小，可设置范围 4Bytes 至 4GB，默认为 8MB。

```
thread_cache_size = 64
```

设置 Thread Cache 池中可以缓存的连接线程最大数量，可设置为 0 至 16384，默认为 0。这个值表示可以重新利用保存在缓存中线程的数量，当断开连接时如果缓存中还有空间，那么客户端的线程将被放到缓存中；如果线程重新被请求，那么请求将从缓存中读取；如果缓存中是空的或者是新的请求，那么这个线程将被重新创建；如果有很多新的线程，增加这个值可以改善系统性能。通过比较 Connections 和 Threads_created 状态的变量，可以看到这个变量的作用。我们可以根据物理内存设置规则如下：1GB 内存我们配置为 8，2GB 内存我们配置为 16，3GB 我们配置为 32，4GB 或 4GB 以上我们给此值为 64 或更大的数值。

```
query_cache_size = 64M
```

指定 MySQL 查询缓冲区的大小。可以通过在 MySQL 控制台观察，如果 `Qcache_lowmem_prunes` 的值非常大，则表明经常出现缓冲不够的情况；如果 `Qcache_hits` 的值非常大，则表明查询缓冲使用得非常频繁。另外，如果该值较小反而会影响效率，那么可以考虑不用查询缓冲。对于 `Qcache_free_blocks`，如果该值非常大，则表明缓冲区中碎片很多。

```
tmp_table_size = 256M
```

设置内存临时表最大值。如果超过该值，则会将临时表写入磁盘，其范围为 1KB 到 4GB。

```
max_connections = 768
```

指定 MySQL 允许的最大连接进程数。如果在访问论坛时经常出现 Too Many Connections 的错误提示，则需要增大该参数值。

```
max_connect_errors = 1000
```

设置每个主机的连接请求异常中断的最大次数，当超过该次数，MySQL 服务器将禁止 host 的连接请求，直到 MySQL 服务器重启或通过 flush hosts 命令清空此 host 的相关信息。此值可设置为 1 至

4, 默认为 10。

```
wait_timeout = 10
```

指定一个请求的最大连接时间, 对于 4GB 左右内存的服务器来说, 可以将其设置为 5 ~ 10。

```
thread_concurrency = 8
```

该参数取值为服务器逻辑 CPU 数量 $\times 2$, 在本例中, 服务器有两个物理 CPU, 而每个物理 CPU 又支持 H.T 超线程, 所以实际取值为 $4 \times 2 = 8$ 。这也是目前双四核主流服务器的配置。

```
skip-networking
```

开启该选项可以彻底关闭 MySQL 的 TCP/IP 连接方式, 如果 Web 服务器是以远程连接的方式访问 MySQL 数据库服务器的, 则不要开启该选项, 否则将无法连接!

```
table_cache = 1024
```

物理内存越大, 设置就越大。默认为 2402, 调到 512 ~ 1024 最佳。

```
innodb_additional_mem_pool_size = 4M
```

默认为 2MB。

```
innodb_flush_log_at_trx_commit = 1
```

设置为 0 就是等到 innodb_log_buffer_size 列队满后再统一储存, 默认为 1。

```
innodb_log_buffer_size = 2M
```

默认为 1MB。

```
innodb_thread_concurrency = 8
```

你的服务器有几个 CPU 就设置为几, 建议用默认设置, 一般为 8。

```
tmp_table_size = 64M
```

设置内存临时表最大值。如果超过该值, 则会将临时表写入磁盘。设置范围为 1KB 至 4GB。

```
read_rnd_buffer_size = 16M
```

read_rnd_buffer_size 设置进行随机读的时候所使用的缓冲区。此参数和 read_buffer_size 所设置的 Buffer 相反, 一个是顺序读的时候使用, 一个是随机读的时候使用。但是两者都是针对于线程的设置, 每个线程都可以产生两种 Buffer 中的任何一个。read_rnd_buffer_size 的默认值 256KB, 最大值 4GB。

值得注意的是:

- 如果 key_reads 太大, 则应该把 my.cnf 中的 key_buffer_size 变大, 保持 key_reads/key_read_requests 至少在 1/100 以上, 越小越好。
- 如果 qcache_lowmem_prunes 很大, 就要增加 query_cache_size 的值。

不过, 很多时候需要具体情况具体分析, 其他参数的变更我们可以等 MySQL 上线稳定一段时间后再根据 status 值进行调整。

附上我们的电子商务网站 MySQL 数据库 (服务器为 DELL R710, 16G 内存, RAID5) 调整后所

运行的配置文件/etc/my.cnf，大家可以根据自己实际的 MySQL 数据库的硬件情况进行调整。配置文件内容如下：

```
[client]
default-character-set=utf8
port = 3306
socket = /tmp/mysql.sock

[mysqld]
user = mysql
port = 3306
socket = /tmp/mysql.sock
basedir = /usr/local/mysql
datadir = /data/mysql/data
log-error = /data/mysql/mysql-error.log
pid-file = /data/mysql/mysql.pid
old-passwords = 1

log_slave_updates = 1
log-bin = /data/mysql/binlog/mysql-bin
binlog_format = mixed
binlog_cache_size = 4M
max_binlog_cache_size = 8M
max_binlog_size = 1G
expire_logs_days = 90
binlog-ignore-db = mysql
binlog-ignore-db = test
binlog-ignore-db = information_schema

key_buffer_size = 384M
sort_buffer_size = 2M
read_buffer_size = 2M
read_rnd_buffer_size = 16M
join_buffer_size = 2M
thread_cache_size = 8
query_cache_size = 32M
query_cache_limit = 2M
query_cache_min_res_unit = 2k
thread_concurrency = 32

table_cache = 614
table_open_cache = 512
open_files_limit = 10240
back_log = 600
max_connections = 5000
max_connect_errors = 6000
external-locking = FALSE

max_allowed_packet = 16M
default-storage-engine = MyISAM
```

```

thread_stack = 192K
transaction_isolation = READ-COMMITTED
tmp_table_size = 256M
max_heap_table_size = 512M

bulk_insert_buffer_size = 64M
myisam_sort_buffer_size = 64M
myisam_max_sort_file_size = 10G
myisam_repair_threads = 1
myisam_recover

long_query_time = 2
slow_query_log
slow_query_log_file = /data/mysql/slow.log
skip-name-resolve
skip-locking
skip-networking

innodb_additional_mem_pool_size = 16M
innodb_buffer_pool_size = 512M
innodb_data_file_path = ibdata1:256M:autoextend
innodb_file_io_threads = 4
innodb_thread_concurrency = 8
innodb_flush_log_at_trx_commit = 2
innodb_log_buffer_size = 16M
innodb_log_file_size = 128M
innodb_log_files_in_group = 3
innodb_max_dirty_pages_pct = 90
innodb_lock_wait_timeout = 120
innodb_file_per_table = 0

[mysqldump]
quick
max_allowed_packet = 64M

[mysql]
no-auto-rehash
Remove the next comment character if you are not familiar with SQL
safe-updates

[myisamchk]
key_buffer_size = 256M
sort_buffer_size = 256M
read_buffer = 2M
write_buffer = 2M

[mysqlhotcopy]
interactive-timeout

```

6.7.4 MySQL 上线后根据 status 状态进行适当优化

MySQL 数据库上线后，可以等其稳定运行一段时间后再根据服务器的 status 状态进行适当优化，我们可以用如下命令列出 MySQL 服务器运行的各种状态值：

```
mysql > show global status;
```

我个人较喜欢的用法是 `show status like '查询值%';`。

1. 慢查询

有时我们为了定位系统中效率比较低下的 Query 语名，需要打开慢查询日志，也就是 Slow Query Log。打开慢查询日志的相关命令如下：

```
mysql > show variables like '%slow%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| log_slow_queries | ON |
| slow_launch_time | 2 |
+-----+-----+
mysql > show global status like '%slow%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| Slow_launch_threads | 0 |
| Slow_queries | 4148 |
+-----+-----+
```

打开慢查询日志可能会对系统性能有一点点影响，如果你的 MySQL 是主-从结构，可以考虑打开其中一台从服务器的慢查询日志，这样既可以监控慢查询，对系统性能影响也会很小。另外，可用 MySQL 自带的命令 `mysqldumpslow` 进行查询。比如，下面的命令可以查出访问次数最多的 20 个 SQL 语句：

```
mysqldumpslow -s c -t 20 host -slow.log
```

2. 连接数

我们如果经常会遇见 MySQL: ERROR 1040: Too many connections 的情况，一种情况是访问量确实很高，MySQL 服务器扛不住了，这个时候就要考虑增加从服务器分散读压力。另外一种情况是 MySQL 配置文件中 `max_connections` 的值过小。来看一个例子，如下所示：

```
mysql > show variables like 'max_connections';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| max_connections | 256 |
+-----+-----+
```

这台 MySQL 服务器的最大连接数是 256，然后查询一下该服务器响应的最大连接数：

```
mysql > show global status like 'Max_used_connections';
```



```

+-----+
| Variable_name | Value |
+-----+
| Max_used_connections | 245 |
+-----+

```

MySQL 服务器过去的最大连接数是 245，没有达到服务器连接数的上限 256，应该不会出现 1040 错误。比较理想的设置是：

$$\text{Max_used_connections} / \text{max_connections} * 100\% \approx 85\%$$

最大连接数占上限连接数的 85% 左右，如果发现比例在 10% 以下，则说明 MySQL 服务器连接数的上限设置得过高了。

3. key_buffer_size

key_buffer_size 是设置 MyISAM 表索引缓存空间的大小，此参数对 MyISAM 表性能影响最大。下面是一台以 MyISAM 为主要存储引擎服务器的配置：

```

mysql> show variables like 'key_buffer_size';
+-----+
| Variable_name | Value |
+-----+
| key_buffer_size | 536870912 |
+-----+

```

从上面的配置可以看出，分配了 512MB 内存给 key_buffer_size。我们再看一下 key_buffer_size 的使用情况：

```

mysql> show global status like 'key_read%';
+-----+
| Variable_name | Value |
+-----+
| Key_read_requests | 27813678764 |
| Key_reads | 6798830 |
+-----+

```

一共有 27 813 678 764 个索引读取请求，有 6 798 830 个请求在内存中没有找到，直接从硬盘读取索引，计算索引未命中缓存的概率：

$$\text{key_cache_miss_rate} = \text{Key_reads} / \text{Key_read_requests} * 100\%$$

比如上面的数据，key_cache_miss_rate 为 0.0244%，4000 个索引读取请求才有一个直接读硬盘，效果已经很好了，key_cache_miss_rate 在 0.1%（即每 1000 个请求有一个直接读硬盘）以下都很好，如果 key_cache_miss_rate 在 0.01% 以下的话，则说明 key_buffer_size 分配得过多，可以适当减少。

MySQL 服务器还提供了 key_blocks_* 参数，如下所示：

```

mysql> show global status like 'key_blocks_u%';
+-----+
| Variable_name | Value |
+-----+

```

```
| Key_blocks_unused      | 0          |
| Key_blocks_used       | 413543     |
+-----+-----+
```

key_blocks_unused 表示未使用的缓存簇 (blocks) 数, key_blocks_used 表示曾经用到的最大的 blocks 数。比如这台服务器, 所有的缓存都用到了, 要么增加 key_buffer_size, 要么就是过度索引, 把缓存占满了。比较理想的设置是:

$$\text{Key_blocks_used} / (\text{Key_blocks_unused} + \text{Key_blocks_used}) * 100\% \approx 80\%$$

4. 临时表

当执行语句时, 关于已经被创造了的隐含临时表的数量, 我们可以用如下命令查询其具体情况:

```
mysql> show global status like 'created_tmp%';
+-----+-----+
| Variable_name      | Value |
+-----+-----+
| Created_tmp_disk_tables | 21197 |
| Created_tmp_files    | 58    |
| Created_tmp_tables   | 1771587 |
+-----+-----+
```

每次创建临时表时, Created_tmp_tables 都会增加, 如果是在磁盘上创建临时表, Created_tmp_disk_tables 也会增加。Created_tmp_files 表示 MySQL 服务创建的临时文件数, 比较理想的配置是:

$$\text{Created_tmp_disk_tables} / \text{Created_tmp_tables} * 100\% \leq 25\%$$

比如上面的服务器 $\text{Created_tmp_disk_tables} / \text{Created_tmp_tables} * 100\% = 1.20\%$, 应该说是相当好了。我们再看一下 MySQL 服务器对临时表的配置:

```
mysql> show variables where Variable_name in ('tmp_table_size', 'max_heap_table_size');
+-----+-----+
| Variable_name      | Value |
+-----+-----+
| max_heap_table_size | 268435456 |
| tmp_table_size      | 536870912 |
+-----+-----+
```

只有 256MB 以下的临时表才能全部放在内存中, 超过的就会用到硬盘临时表。

5. 打开表的情况

Open_tables 表示打开表的数量, Opened_tables 表示打开过的表数量, 我们可以用如下命令查看其具体情况:

```
mysql> show global status like 'open%tables%';
+-----+-----+
| Variable_name      | Value |
+-----+-----+
| Open_tables        | 919   |
| Opened_tables      | 1951  |
```

```
+-----+-----+
```

如果 `Opened_tables` 数量过大, 说明配置中 `table_cache` (MySQL 5.1.3 之后这个值叫做 `table_open_cache`) 的值可能太小。我们查询一下服务器 `table_cache` 值:

```
mysql> show variables like 'table_cache';
```

```
+-----+-----+
```

```
| Variable_name | Value |
```

```
+-----+-----+
```

```
| table_cache | 2048 |
```

```
+-----+-----+
```

比较合适的值为:

```
Open_tables / Opened_tables * 100% >= 85%
```

```
Open_tables / table_cache * 100% <= 95%
```

6. 进程使用情况

如果我们在 MySQL 服务器的配置文件中设置了 `thread_cache_size`, 当客户端断开之时, 服务器处理此客户请求的线程将会缓存起来以响应下一个客户而不是销毁 (前提是缓存数未达上限)。`Threads_created` 表示创建过的线程数, 我们可以用如下命令查看:

```
mysql> show global status like 'Thread%';
```

```
+-----+-----+
```

```
| Variable_name | Value |
```

```
+-----+-----+
```

```
| Threads_cached | 46 |
```

```
| Threads_connected | 2 |
```

```
| Threads_created | 570 |
```

```
| Threads_running | 1 |
```

```
+-----+-----+
```

如果发现 `Threads_created` 的值过大的话, 表明 MySQL 服务器一直在创建线程, 这也是比较耗费资源的, 可以适当增大配置文件中 `thread_cache_size` 的值。查询服务器 `thread_cache_size` 配置, 如下所示:

```
mysql> show variables like 'thread_cache_size';
```

```
+-----+-----+
```

```
| Variable_name | Value |
```

```
+-----+-----+
```

```
| thread_cache_size | 64 |
```

```
+-----+-----+
```

示例中的 MySQL 服务器还是挺健康的。

7. 查询缓存 (query cache)

它涉及的主要有两个参数, `query_cache_size` 是设置 MySQL 的 Query Cache 大小, `query_cache_type` 是设置使用查询缓存的类型, 我们可以用如下命令查看其具体情况:

```
mysql> show global status like 'qcache%';
```

```

+-----+-----+
| Variable_name      | Value      |
+-----+-----+
| Qcache_free_blocks | 22756      |
| Qcache_free_memory | 76764704   |
| Qcache_hits        | 213028692  |
| Qcache_inserts     | 208894227  |
| Qcache_lowmem_prunes | 4010916    |
| Qcache_not_cached  | 13385031   |
| Qcache_queries_in_cache | 43560      |
| Qcache_total_blocks | 111212     |
+-----+-----+

```

MySQL 查询缓存变量的相关解释如下。

- `Qcache_free_blocks`: 缓存中相邻内存块的个数。数目大说明可能有碎片。`FLUSH QUERY CACHE` 会对缓存中的碎片进行整理, 从而得到一个空闲块。
- `Qcache_free_memory`: 缓存中的空闲内存。
- `Qcache_hits`: 多少次命中。通过这个参数可以查看到 Query Cache 的基本效果。
- `Qcache_inserts`: 插入次数, 每次插入一个查询时就增加 1。命中次数除以插入次数就是命中率。
- `Qcache_lowmem_prunes`: 多少条 Query 因为内存不足而被清除出 Query Cache。通过 `Qcache_lowmem_prunes` 和 `Qcache_free_memory` 相互结合, 能够更清楚地了解到系统中 Query Cache 的内存大小是否真的足够, 是否非常频繁地出现因为内存不足而有 Query 被换出的情况。
- `Qcache_not_cached`: 不适合进行缓存的查询数量, 通常是由于这些查询不是 `SELECT` 语句或用了 `now()` 之类的函数。
- `Qcache_queries_in_cache`: 当前缓存的查询 (和响应) 数量。
- `Qcache_total_blocks`: 缓存中块的数量。

我们再查询一下服务器上关于 `query_cache` 的配置命令如下:

```

mysql> show variables like 'query_cache%';
+-----+-----+
| Variable_name      | Value      |
+-----+-----+
| query_cache_limit   | 2097152    |
| query_cache_min_res_unit | 4096      |
| query_cache_size    | 203423744  |
| query_cache_type    | ON         |
| query_cache_wlock_invalidate | OFF       |
+-----+-----+

```

各字段的解释如下所示:

- `query_cache_limit`: 超过此大小的查询将不缓存。
- `query_cache_min_res_unit`: 缓存块的最小值。
- `query_cache_size`: 查询缓存大小。
- `query_cache_type`: 缓存类型, 决定缓存什么样的查询, 示例中表示不缓存 `select sql_no_`

cache 查询。

- `query_cache_wlock_invalidate`: 表示当有其他客户端正在对 MyISAM 表进行写操作时, 读请求是要等 WRITE LOCK 释放资源后再查询还是允许直接从 Query Cache 中读取结果, 默认为 OFF (可以直接从 Query Cache 中取得结果)。

`query_cache_min_res_unit` 的配置是一柄“双刃剑”, 默认是 4KB, 设置值大对大数据查询有好处, 但如果你的查询都是小数据查询, 就容易造成内存碎片和浪费。

查询缓存碎片率 = $\text{Qcache_free_blocks} / \text{Qcache_total_blocks} * 100\%$

如果查询缓存碎片率超过 20%, 可以用 FLUSH QUERY CACHE 整理缓存碎片, 或者试试减小 `query_cache_min_res_unit`, 如果你的查询都是小数据量的话。

查询缓存利用率 = $(\text{query_cache_size} - \text{Qcache_free_memory}) / \text{query_cache_size} * 100\%$

查询缓存利用率在 25% 以下的话说明 `query_cache_size` 设置得过大, 可适当减小; 查询缓存利用率在 80% 以上而且 `Qcache_lowmem_prunes > 50` 的话则说明 `query_cache_size` 可能有点小, 要不就是碎片太多。

查询缓存命中率 = $(\text{Qcache_hits} - \text{Qcache_inserts}) / \text{Qcache_hits} * 100\%$

示例服务器中的查询缓存碎片率等于 20.46%, 查询缓存利用率等于 62.26%, 查询缓存命中率等于 1.94%, 说明命中率很差, 可能写操作比较频繁吧, 而且可能有些碎片。

8. 排序使用情况

它表示系统中对数据进行排序时所使用的 Buffer, 我们可以用如下命令查看:

```
mysql> show global status like 'sort%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| Sort_merge_passes | 29 |
| Sort_range | 37432840 |
| Sort_rows | 9178691532 |
| Sort_scan | 1860569 |
+-----+-----+
```

`Sort_merge_passes` 包括如下步骤: MySQL 首先会尝试在内存中做排序, 使用的内存大小由系统变量 `sort_buffer_size` 来决定, 如果它不够大则把所有的记录都读到内存中, 而 MySQL 则会把每次在内存中排序的结果存到临时文件中, 等 MySQL 找到所有记录之后, 再把临时文件中的记录做一次排序。这次再排序就会增加 `sort_merge_passes`。实际上, MySQL 会用另一个临时文件来存储再次排序的结果, 所以我们通常会看到 `sort_merge_passes` 增加的数值是建临时文件数的两倍。因为用到了临时文件, 所以速度可能会比较慢, 增大 `sort_buffer_size` 会减少 `sort_merge_passes` 和创建临时文件的次数, 但盲目地增大 `sort_buffer_size` 并不一定能提高速度。

9. 文件打开数 (open_files)

我们在处理 MySQL 故障时, 发现当 `open_files` 大于 `open_files_limit` 值时, MySQL 数据库就会发生卡住的现象, 导致 Apache 服务器打不开相应页面。这个问题大家在工作中应注意, 我们可以用如下命令查看其具体情况:

```
mysql> show global status like 'open_files';
```

```

+-----+-----+
| Variable_name | Value |
+-----+-----+
| Open_files | 1410 |
+-----+-----+
mysql> show variables like 'open_files_limit';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| open_files_limit | 4590 |
+-----+-----+

```

比较合适的设置是： $\text{Open_files} / \text{open_files_limit} * 100\% \leq 75\%$ 。

很多时候我们会发现，通过参数设置进行性能优化所带来的性能提升，并不如许多人想象的那样会产生质的飞跃，除非是之前的设置存在严重不合理的情况。我们不能将性能调优完全依托于通过 DBA 在数据库上线后进行参数调整，而应该在系统设计和开发阶段就尽可能地减少性能问题。

6.7.5 MySQL 数据库的可扩展性架构方案

如果凭借 MySQL 的优化仍无法顶住压力，这个时候我们就必须考虑 MySQL 的可扩展性架构了（也有人将它说成是 MySQL 集群），它有以下明显的优势：

- 成本低，很容易通过价格低廉的 PC Server 搭建出一个处理能力非常强大的计算机集群。
- 不太容易遇到瓶颈，因为很容易通过添加主机来增加处理能力。
- 单节点故障对系统的整体影响较小。

目前可行的方案如下：

（1）MySQL Cluster

其特点为可用性非常高，性能非常好。每份数据至少可在不同主机上存一份拷贝，且冗余数据拷贝实时同步。但它的维护非常复杂，存在部分 Bug，目前还不适合比较核心的线上系统，所以暂不推荐。

（2）DRBD 磁盘网络镜像方案

其特点为软件功能强大，数据可在底层块设备级别跨物理主机镜像，且可根据性能和可靠性要求配置不同级别的同步。I/O 操作会保持顺序，可满足数据库对数据一致性的苛刻要求。但非分布式文件系统环境无法支持镜像数据同时可见，性能和可靠性两者相互矛盾，无法适用于性能和可靠性要求都比较苛刻的环境，维护成本高于 MySQL Replication。另外，DRBD 是官方推荐的可用于 MySQL 的高可用方案之一，大家可根据实际环境来考虑是否部署。

（3）MySQL Replication

在工作中，此种 MySQL 高可用、高扩展性架构也是用得最多的，我也推荐此种方案。下一节我向为大家介绍几种 MySQL Replication 的高可用架构方案。

6.7.6 MySQL 数据库的 Replication 高可用架构

我们工作中经常用到的 MySQL 数据库的 Replication 高可用架构方案有如下几种：

(1) “MySQL 主从 Replication” 方案

这是最常见也是最简单的一种架构，就是一台 MySQL 服务器 (slave) 从另一台 MySQL 服务器 (Master) 上将日志复制过去，然后再解析日志并应用到自身。一个复制环境仅仅需要两台运行有 MySQL 的 Server 主机即可，我的许多项目方案和现在公司的网站都是采用这种架构，维护起来也非常方便。

(2) 常规复制架构 (Master-Slaves)

在实际应用场景中，90% 以上的 MySQL 复制都是从一个 Master 复制到一个或多个 Slave 的架构模式上，主要用于读压力大的应用数据库端廉价扩展解决方案。因为只要 Master 和 Slave 的压力不是太大（尤其是 Slave 端的压力），异步复制的延时一般都很少很少。尤其是从 Slave 端的复制方式改成两个线程处理以后，进一步降低了 Slave 端的延时问题。而带来的效益是，对于数据实时性要求不是特别 Critical 的应用，只需要通过廉价的 PC Server 来扩展 Slave 的数量，就可将读压力分散到多台 Slave 的机器上，也就可以通过分散单台数据库服务器的读压力来解决数据库端的读性能瓶颈，毕竟在大多数数据库应用系统中读的压力比写的压力还是要大很多。这在很大程度上解决了很多中小型网站数据库的压力瓶颈问题，甚至有些大型网站也在使用类似的方案。

(3) Dual Master 复制架构 (Master-Master)

有时候，我们需要在一些特定的场景下进行 Master 的切换，如果在 Master 端进行一些特别的维护操作时要停止 MySQL 服务。那我们该如何保证所有节点数据的一致性呢？其实可以搭建 Dual Master 环境来处理。何谓 Dual Master 环境？实际上就是两个 MySQL Server 互相视对方为自己的 Master，然后把自己作为对方的 Slave 来进行复制。这样，任何一方所做的变更，都可以通过复制应用到另外一方的数据库中。通过 Dual Master 复制架构，能够避免一个问题，即我们进行常规的维护操作时因停机所带来的重新搭建 Replication 环境的问题，因为两个 MySQL Server 已经记录了自己当前复制到对方的什么位置，在系统搭建之后，它们会自动从之前的位置开始继续复制，不需要人为干预，大大节省了维护成本。不仅如此，Dual Master 复制架构和一些第三方的 HA 管理软件相结合，还可以在当使用的 Master 出现异常无法提供服务时，非常迅速地自动切换到另外一端来提供相应的服务，减少异常情况下带来的停机时间，也不需要人为干预。

(4) 级联复制架构 (Master-Slaves-Slaves)

在有些应用场景中，也许读写压力差别比较大，当读压力特别大时，一个 Master 可能需要 10 台甚至更多 Slave 才能支撑读的压力。这时候，Master 就会比较吃力，仅仅是连接上来的 Slave I/O 线程已经比较多了，要是写的压力稍微再大一点，Master 端就会因为复制消耗更多的资源，很容易造成复制的延时。遇到这种情况该如何解决呢？这时候我们可以利用 MySQL 在 Slave 端记录因复制而产生变更的 Binary Log 信息，也就是打开 `--log-slave-update` 选项，然后通过二级（或者是更多级别）复制来减少 Master 端因为复制所带来的压力。从第一级 Slave 上进行复制的 Slave，我们称为第二级 Slave 集群，这种架构则称之为 Master-Slave-Slave 架构。通过这种多层级联复制的架构，很容易就解决了 Master 端因为附属 Slave 太多而造成的瓶颈风险。

(5) Dual Master 与级联复制结合架构 (Master-Master-Slaves)

级联复制在一定程度上解决了 Master 因为所附属的 Slave 过多而造成的瓶颈问题，但是它并不能解决人工维护和出现异常需求要进行切换时可能存在的重新搭建 Replication 的问题。这就很自然地引出了 Dual Master 与级联复制结合的 Replication 架构，我称之为 Master-Master-Slaves 架构，这种

架构最大的好处就是既可以避免主 Master 的写操作受到 Slave 集群复制所带来的影响，同时主 Master 需要切换的时候也基本不会出现重搭 Replication 的情况。但是，这个架构也有一个弊端，那就是备用的 Master 也有可能成为瓶颈，因为如果后面的 Slave 集群比较大的话，备用 Master 可能会因为有过多的 Slave I/O 线程请求而成为瓶颈。当该备用 Master 不提供任何读服务时，瓶颈出现的可能性并不是特别高，如果出现瓶颈，也可以在备用 Master 后面再进行级联复制，架设多层 Slave 集群。当然，级联复制的级别越多，Slave 集群可能出现的数据延时也会更明显，所以考虑使用多层级联复制之前，也需要评估数据延时效对应用系统的影响。

在实际应用场景中，MySQL Replication 是使用最为广泛的一种提高系统扩展性的设计手段。众多的 MySQL 使用者通过 Replication 功能提升系统的扩展性之后，通过简单地增加价格低廉的硬件设备成倍甚至成数量级地提高了原有系统的性能，是广大 MySQL 中低端使用者非常喜爱的功能之一，也是很多 MySQL 使用者选择 MySQL 最为重要的理由。

6.7.7 MySQL Cluster 集群配置方案

在为某证券公司设计其 OA 架构时，初期客户预计是 30 万用户同时在线，然而在项目的实施过程中，客户又提出了 50 万用户同时在线的需求，而且都会对数据库读写操作。这样一来，初始的设计方案便满足不了客户的要求了（初始方案是 Master-Slaves，读写分离），所以我们打算试用 MySQL Cluster 方案。MySQL Cluster 是 MySQL 适合于分布式计算环境的高实用、高冗余版本。它采用了 NDB Cluster 存储引擎，允许在 1 个 Cluster 中运行多个 MySQL 服务器。在 MySQL 5.0 及以上的二进制版本，以及与最新的 Linux 版本兼容的 RPM 中提供了该存储引擎。

1. MySQL Cluster 概述

MySQL Cluster 是一种技术，该技术允许在无共享的系统中部署“内存中”数据库的 Cluster。通过无共享体系结构，系统能够使用廉价的硬件，而且对软硬件无特殊要求。此外，由于每个组件有自己的内存和磁盘，所以不存在单点故障。

MySQL Cluster 由一组计算机构成，每台计算机上均运行着多种进程，包括 MySQL 服务器、NDB Cluster 的数据节点、管理服务器，以及（可能）专门的数据访问程序。

所有的这些节点构成了一个完整的 MySQL 集群体系。数据保存在“NDB 存储服务器”的存储引擎中，表（结构）则保存在“MySQL 服务器”中。应用程序通过“MySQL 服务器”访问这些数据表，集群管理服务器通过管理工具（ndb_mgmd）来管理“NDB 存储服务器”。

通过将 MySQL Cluster 引入开放源码世界，MySQL 为所有需要它的人员提供了具有高可用性、高性能和可缩放性的 Cluster 数据管理。

2. MySQL Cluster 的基本概念

“NDB”是一种“内存中”的存储引擎，它具有可用性高和数据一致性好的特点。MySQL Cluster 能够使用多种故障切换和负载均衡选项来配置 NDB 存储引擎，但在 Cluster 级别的存储引擎上做这个是最简单的。目前，MySQL Cluster 的 Cluster 可独立于 MySQL 服务器进行配置。在 MySQL Cluster 中，Cluster 的每个部分都被视为 1 个节点。

管理（MGM）节点：这类节点的作用是管理 MySQL Cluster 内的其他节点，如提供配置数据、

启动并停止节点、运行备份等。由于这类节点负责管理其他节点的配置，所以应在启动其他节点之前首先启动这类节点。MGM节点是用命令“ndb_mgmd”来启动的。

数据节点：这类节点用于保存 Cluster 的数据。数据节点的数目与副本的数目相关，是片段的倍数。假设有两个副本，每个副本有两个片段，那么就有4个数据节点。不过没有必要设置多个副本。数据节点是用命令“ndbd”来启动的。

SQL节点：这是用来访问 Cluster 数据的节点。对于 MySQL Cluster 来说，客户端节点是使用 NDB Cluster 存储引擎的传统 MySQL 服务器。通常，SQL节点是使用命令“mysqld-ndbcluster”来启动的，或者将“ndbcluster”添加到“my.cnf”后使用“mysqld”启动。

注意 在很多情况下，术语“节点”是用来指计算机的，但在讨论 MySQL Cluster 时，它表示的是进程。在单台计算机上可以有任意数目的节点，为此，我们采用术语“Cluster 主机”来表示。

管理服务器（MGM节点）：负责管理 Cluster 配置文件和 Cluster 日志。Cluster 中的每个节点从管理服务器上检索配置数据，并请求确定管理服务器所在位置的方式。当数据节点内出现新的事件时，节点将关于这类事件的信息传输到管理服务器上，然后，又将这类信息写入 Cluster 日志。

此外，可以有任意数目的 Cluster 客户端进程或应用程序。它们分为两种类型，即标准 MySQL 客户端和管理客户端。

标准 MySQL 客户端：对于 MySQL Cluster 来说，它们与标准的（非 Cluster 类）MySQL 没有区别。换句话讲，也就是能够从用 PHP、Perl、C、C++、Java、Python、Ruby 等编写的现有 MySQL 应用程序上访问 MySQL Cluster。

管理客户端：这类客户端与管理服务器相连，并提供了启动和停止节点、启动和停止消息跟踪（仅调试版本）、显示节点版本和状态、启动和停止备份等命令。

MySQL Cluster 的架构示意图（出自 MySQL 官方文档手册）如图 6-10 所示。

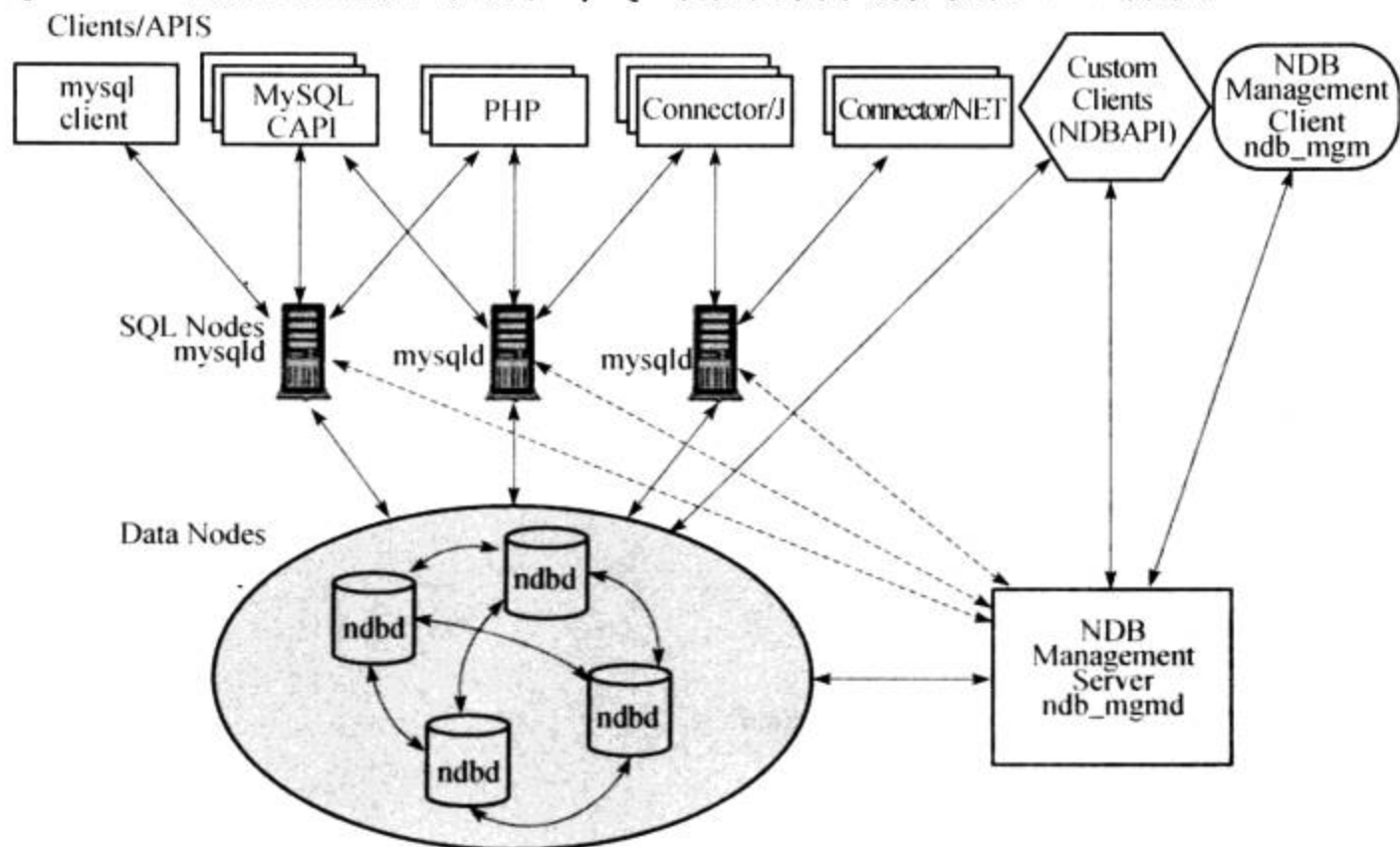


图 6-10 MySQL Cluster 架构示意图

3. 做好准备

(1) 准备服务器（内部测试环境，非正式线上环境）

现在，我们计划建立有 5 个节点的 MySQL Cluster 体系，因此需要用到 5 台机器，分别做如下用途，如表 6-4 所示。

(2) 注意事项及其他

每个节点的操作系统都是 Centos5.2 x86_64，为了不让大家混淆，在下面的描述中我将使用主机名来表示机器。由于 MySQL Cluster 采用的是 TCP/IP 方式连接，并且节点之间的数据传输没有加密，因此这个体系最好只在单独的子网中运行。另外考虑到

表 6-4 MySQL Cluster 测试实验服务器 IP 分配表

用途	主机名	IP
管理节点	ndb_mgmd	192.168.5.101
数据节点 1	ndb1	192.168.5.102
数据节点 2	ndb2	192.168.5.103
sql 节点 1	mysqld1	192.168.5.104
sql 节点 2	mysqld2	192.168.5.105

传输的速率问题，强烈建议不要跨越公网使用这个体系。为了测试环境的准确性，我这里所有机器的 hosts 文件都做好了相应的对应关系，所需软件推荐使用 mysql-max-5.0.24，下载地址为：<http://www.filewatcher.com/?q=mysql-max-5.0.24-linux-i686.tar.gz>。另外申明一点，以下均为内部测试环境，暂时未用于线上环境，有兴趣的朋友可以关注一下，了解一下 MySQL Cluster 的搭建环境。

4. 开始安装

1) 在每个 SQL 节点计算机上都采用 MySQL 用户来运行 Cluster，因此可执行如下命令添加相关用户（如果已经存在则略过，且用 root 用户执行）：

```
/usr/sbin/groupadd mysql
/usr/sbin/useradd -g mysql mysql
```

假设已经下载了 MySQL 可直接使用的二进制安装包，且放在了 /root 下。

2) SQL 节点和存储节点（NDB 节点）的安装如下（另外 4 个机器重复执行以下步骤）：

```
cd /root/
tar zxf mysql-max-5.0.24-linux-i686.tar.gz
mv mysql-max-5.0.24-linux-i686 /usr/local/mysql/
cd /usr/local/mysql/
./configure --prefix=/usr/local/mysql
./scripts/mysql_install_db
chown -R mysql:mysql /usr/local/mysql/
```

3) 配置 SQL 节点，命令如下：

```
root# vim /etc/my.cnf
```

然后输入如下内容：

```
[mysqld]
basedir          = /usr/local/mysql/
datadir          = /usr/local/mysql/data/
user             = mysql
port             = 3306
socket           = /tmp/mysql.sock
ndbcluster
```

```
ndb - connectstring = 192.168.5.101
[mysql_cluster]
ndb - connectstring = 192.168.5.101
```

4) 配置存储节点 (NDB 节点), 命令如下:

```
root# vim /etc/my.cnf
```

然后输入如下内容:

```
[mysqld]
ndbcluster
ndb - connectstring = 192.168.5.101
[mysql_cluster]
ndb - connectstring = 192.168.5.101
```

5) 安装管理节点, 命令如下:

```
root# cd /root/
root# tar xzf mysql-max-5.0.24-linux-i686.tar.gz
root# mkdir /usr/local/mysql/
root# mkdir /usr/local/mysql/data/
root# cd mysql-max-5.0.24-linux-i686/bin/
root# cp ndb_mgm* /usr/local/mysql
root# chown -R mysql:mysql /usr/local/mysql
```

6) 配置管理节点, 命令如下:

```
root# vi /usr/local/mysql/config.ini
```

然后输入如下内容:

```
[ndbd default]
NoOfReplicas = 1
[tcp default]
portnumber = 3306
```

设置管理节点服务器的命令如下:

```
[ndb_mgmd]
hostname = 192.168.5.101
```

在 MGM 上保存日志的目录。

```
datadir = /usr/local/mysql/data/
```

设置存储节点服务器 (NDB 节点), 命令如下:

```
[ndbd]
hostname = 192.168.5.102
datadir = /usr/local/mysql/data/
```

设置第二个 NDB 节点, 命令如下:

```
[ndbd]
hostname = 192.168.5.103
datadir = /usr/local/mysql/data/
```

设置 SQL 节点服务器，命令如下：

```
[mysqld]
hostname=192.168.5.104
```

#设置第二个 SQL 节点，命令如下：

```
[mysqld]
hostname=192.168.5.105
```

注意 [tcp default] 这项要写成 portnumber = 3306。

5. 启动 MySQL Cluster

较为合理的启动顺序是，先启动管理节点服务器，然后启动存储节点服务器，最后才启动 SQL 节点服务器。在管理节点服务器上，执行以下命令启动 MGM 节点进程：

```
/usr/local/mysql/ndb_mgmd -f /usr/local/mysql/config.ini
```

必须用参数 -f 或 --config-file 告诉 ndb_mgm 配置文件 config.ini 文件所在的位置，默认是在与 ndb_mgmd 相同的目录下。

在每台存储节点服务器上，如果是第一次启动 NDBD 进程的话，必须先执行以下命令：

```
/usr/local/mysql/bin/ndbd --initial
```

注意 仅应在首次启动 NDBD 时，或者在备份/恢复数据或配置文件发生变化且重启 NDBD 时才使用 -initial 参数。因为该参数会使节点删除由早期 NDBD 实例创建的、用于恢复的任何文件，包括用于恢复的日志文件。

如果不是第一次启动，直接运行如下命令即可：

```
/usr/local/mysql/bin/ndbd
```

最后，运行以下命令启动 SQL 节点服务器：

```
/usr/local/mysql/bin/mysqld_safe --defaults-file=/etc/my.cnf &
```

如果一切顺利，也就是启动过程中没有出现任何错误信息，那么就在管理节点服务器上运行如下命令：

```
/usr/local/mysql/ndb_mgm
-- NDB Cluster -- Management Client --
ndb_mgm> show
Cluster Configuration
-----
[ndbd(NDB)] 2 node(s)
id=2 @192.168.5.102 (Version: 5.0.24, Nodegroup: 0, Master)
id=3 @192.168.5.103 (Version: 5.0.24, Nodegroup: 1)

[ndb_mgmd(MGM)] 1 node(s)
id=1 @192.168.5.101 (Version: 5.0.24)
```



```
[mysqld(API)] 2 node(s)
id=4 @192.168.5.104 (Version: 5.0.24)44
id=5 @192.168.5.105 (Version: 5.0.24)
```

6. 环境测试

与没有使用 Cluster 的 MySQL 相比, 在 MySQL Cluster 内操作数据的方式没有太大的区别。执行这类操作时应记住两点:

- 表必须用 ENGINE = NDB 或 ENGINE = NDBCLUSTER 选项创建, 或用 ALTER TABLE 选项更改, 使用 NDB Cluster 存储引擎在 Cluster 内复制它们。如果使用 mysqldump 的输出从已有的数据库中导入表, 可在文本编辑器中打开 SQL 脚本, 并将该选项添加到任何表创建语句中, 或者用这类选项中的一个替换任何已有的 ENGINE (或 TYPE) 选项。
- 每个 NDB 表必须有一个主键。如果在创建表时用户未定义主键, NDB Cluster 存储引擎会自动生成隐含的主键。

注意 该隐含键也将占用空间, 就像任何其他的表索引一样。由于没有足够的内存来容纳这些自动创建的键, 所以很容易出现问题。

现在我们开始进行环境测试工作了, 步骤如下:

1) 确认 NDB 引擎是否已经正常工作。

在 mysqld1 节点上, 用 root 在 test 数据库上新建一个 t1 的表, 如下所示:

```
mysql> use test;
Database changed
mysql> show tables;
Empty set (0.11 sec)
mysql> create table t1(a int) engine=ndb;
Query OK, 0 rows affected (0.74 sec)
mysql> insert into t1 values(100);
Query OK, 1 row affected (0.32 sec)
```

在 mysqld2 节点上, 用 root 查看效果, 如下所示:

```
mysql> show tables;
+-----+
| Tables_in_test |
+-----+
| t1              |
+-----+
1 row in set (0.00 sec)
```

2) 检测冗余环境的单点故障问题, 冗余环境的单点故障如下所示:

- 模拟 NDB 节点崩溃。
- 模拟 SQL 节点崩溃。

由于管理节点是最容易控制的, 所以可以将配置文件和两个可执行程序 (ndb_mgmd 和 ndb_mgm) 存放在多台机器上, 无需过多考虑单点故障问题。

3) 进行性能测试。

后期内部测试环境中我们增加了两台 NDBD 服务器，即集群方案采用的是 1 个 ndbd_mgmd + 2 个 mysqld + 4 台 NDBD 机器（以 4 台机器作 NDBD 的好处是如果其中两台服务器同时宕机了，也有一定几率不损失数据。事实上，在生产环境下，两台服务器同时宕机的几率是微乎其乎的）。目前公司测试组的同事在进行压力测试和性能测试，尝试进行大量数据的写入和插入等操作。初期打算内部测试 3 个月，但在测试过程中，发现 MySQL Cluster 维护起来非常麻烦，需要大量的人力，远不如 MySQL Replication 方案简单。

7. 安全关闭

要想关闭 Cluster，可在 MGM 节点所在的机器上，在 SHELL 中简单地输入如下命令：

```
/usr/local/mysql/ndb_mgm -e shutdown
```

运行以下命令关闭 SQL 节点的 mysqld 服务：

```
/usr/local/mysql/bin/mysqladmin -uroot shutdown
```

值得一说的是，虽然 MySQL Cluster 目前还存在着不少 Bug，但在高可用和写负载均衡上它的确是性价比很高的解决方案，如果对性能要求不高可尝试一下这种架构。MySQL Cluster 的核心在于 NDB Cluster 存储引擎，它不仅对数据进行了水平切分，还对数据进行了跨节点冗余。既解决了数据库的扩展问题，同时也在很大程度上提高了数据整体的高可用性。考虑到 MySQL Cluster 目前还存在 Bug 问题，且维护成本过大，所以目前我仅仅将其用于内部环境和测试环境（没有将这种技术用于线上环境），但我相信 MySQL Cluster 会越来越成熟、稳定、高效的，让我们共同期待吧！

参考文档和文献：

《MySQL 性能调优与架构设计》，简朝阳，电子工业出版社出版

谭俊青的 MySQL 实验室：<http://www.mysqlab.net/>

<http://vdata.blog.51cto.com/275084/60184>

6.7.8 生产环境下的 MySQL 数据库主从 Replication 同步

MySQL 的主从 Replication 同步（又叫主从复制）是一个很成熟的架构，我的许多线上环境都是采用的这种方案，它的优点如下：

- 在从服务器上可以执行查询工作（即我们常说的读功能），降低主服务器压力。
- 在从服务器上进行备份，避免备份期间影响主服务器服务。
- 当主服务器出现问题时，可以切换到从服务器。

所以我在项目部署和实施过程中经常会采用这种方案。鉴于生产环境下对 MySQL 严谨性的要求，推荐采用 MySQL 源码编译的方法。以下内容我曾在博客中阐述过，并且收到了许多热心网友中肯的意见，我根据大家的意见对此进行了 4 次修改。如果大家在工作有相应的需求，可以参考之。

安装 MySQL 之前建议在服务器上安装基础库文件，命令如下：

```
yum -y install gcc gcc-c++ autoconf libjpeg libjpeg-devel libpng libpng-devel freetype free-
```

```
type -devel libxml2 libxml2 -devel zlib zlib -devel glibc glibc -devel glib2 glib2 -devel bzip2 bzip2
-devel ncurses ncurses -devel curl curl -devel e2fsprogs e2fsprogs -devel krb5 krb5 -devel libidn
libidn -devel openssl openssl -devel
yum groupinstall "Development Tools" "Development Libraries" -yt
```

由于服务器采用的是最小化安装，这里建议也安装一下开发工具和开发库，防止源码编译安装MySQL时报错。

MySQL 数据库涉及的文件及对应目录：

MySQL 的安装位置/usr/local/webserver/mysql/

my.cnf 配置文件/usr/local/webserver/mysql/my.cnf

MySQL 数据库位置/data/data/

记得将这些目录及文件的属主和属组都给予 mysql 用户组及用户，即 `chown -R mysql:mysql` 目录。

下面介绍一下工作环境。

主数据库：192.168.1.106

从数据库：192.168.1.107

操作系统：RHEL5.5 x86_64

服务器类型：HP 580G5，双四核 XeonE 5520，32GB 的内存，将6块300G SAS300G 做成 RAID1 +0

我用的 MySQL 版本为 `mysql-5.5.3-m3.tar.gz`，下载地址为：`wget http://blog.s135.com/soft/linux/nginx_php/mysql/mysql-5.5.3-m3.tar.gz`。目前，我的许多开发环境都是 MySQL-5.5.11，这也是目前最新的版本。我这里推荐采用的还是稳定的 MySQL-5.5.3-m3 版本。

MySQL5.5.3 源码编译过程如下所示：

```
/usr/sbin/groupadd mysql
/usr/sbin/useradd -g mysql mysql
tar zxvf mysql-5.5.3-m3.tar.gz
cd mysql-5.5.3-m3/
./configure --prefix=/usr/local/webserver/mysql/ --enable-assembler --with-extra-char-sets=complex --enable-thread-safe-client --with-big-tables --with-readline --with-ssl --with-embedded-server --enable-local-infile --with-plugins=innobase
make && make install
```

记得将目录的属组和属主都给予 MySQL，命令如下：

```
chown -R mysql:mysql /usr/local/webserver/mysql
chown -R mysql:mysql /data/data/
```

MySQL 配置文件/usr/local/webserver/mysql/my.cnf 的内容如下：

```
[client]
character-set-server = utf8
port = 3306
mysql = /tmp/mysql.sock

[mysqld]
character-set-server = utf8
replicate-ignore-db = mysql
```

```
replicate-ignore-db = test
replicate-ignore-db = information_schema
user = mysql
port = 3306
socket = /tmp/mysql.sock
basedir = /usr/local/webserver/mysql
datadir = /data/data/
log-error = /data/data/mysql_error.log
pid-file = /data/data/mysql.pid
open_files_limit = 10240
back_log = 600
max_connections = 5000
max_connect_errors = 6000
table_cache = 614
external-locking = FALSE
max_allowed_packet = 32M
sort_buffer_size = 1M
join_buffer_size = 1M
thread_cache_size = 300
query_cache_size = 512M
query_cache_limit = 2M
query_cache_min_res_unit = 2k
default-storage-engine = MyISAM
thread_stack = 192K
transaction_isolation = READ-COMMITTED
tmp_table_size = 246M
max_heap_table_size = 246M
long_query_time = 3
log-slave-updates
log-bin = /data/data/binlog
binlog_cache_size = 4M
binlog_format = MIXED
max_binlog_cache_size = 8M
max_binlog_size = 1G
relay-log-index = /data/data/relaylog
relay-log-info-file = /data/data/relaylog
relay-log = /data/data/relaylog
expire_logs_days = 30
key_buffer_size = 256M
read_buffer_size = 1M
read_rnd_buffer_size = 16M
bulk_insert_buffer_size = 64M
myisam_sort_buffer_size = 128M
myisam_max_sort_file_size = 10G
myisam_repair_threads = 1
myisam_recover

interactive_timeout = 120
wait_timeout = 120
```



```

skip-name-resolve
slave-skip-errors = 1032,1062,126,1114,1146,1048,1396

server-id = 1
binlog-do-db=yuhongchun

innodb_additional_mem_pool_size = 16M
innodb_buffer_pool_size = 512M
innodb_data_file_path = ibdata1:256M:autoextend
innodb_file_io_threads = 4
innodb_thread_concurrency = 8
innodb_flush_log_at_trx_commit = 2
innodb_log_buffer_size = 16M
innodb_log_file_size = 128M
innodb_log_files_in_group = 3
innodb_max_dirty_pages_pct = 90
innodb_lock_wait_timeout = 120
innodb_file_per_table = 0

[mysqldump]
quick
max_allowed_packet = 32M

[mysql]
socket = /tmp/mysql.sock
function start mysql
}

function kill_mysql()
{
    kill -9 $(ps -ef | grep 'bin/mysqld_safe' | grep 3306 | awk '{printf $2}')
    kill -9 $(ps -ef | grep 'libexec/mysqld' | grep 3306 | awk '{printf $2}')
}

if [ "$1" = "start" ]; then
    function_start_mysql
elif [ "$1" = "stop" ]; then
    function_stop_mysql
elif [ "$1" = "restart" ]; then
    function_restart_mysql
elif [ "$1" = "kill" ]; then
    function_kill_mysql
else
    printf "Usage: mysqld {start&#124;stop&#124;restart&#124;kill}\n"
fi

```

这里为了调试方便，将 MySQL 数据库的超级管理员密码设置得非常简单，如果是线上维护，建议还是将其设置得复杂且长一些，避免密码很容易就被猜出来（注意安全）。

然后，在两台机器上分别执行如下命令：

```
/usr/local/webserver/mysql/bin/mysql_install_db --basedir=/usr/local/webserver/mysql
--datadir=/data/data --user=mysql
```

以 mysql 用户账号的身份建立数据，即让 /data/data/ 成为 MySQL 存放数据的目录。

启动 MySQL 数据库，命令如下所示：

```
nohup /usr/local/webserver/mysql/bin/mysqld_safe
--defaults-file=/usr/local/webserver/mysql/my.cnf &
```

下面介绍一下主从同步的过程。

1. 设置主库

1) 修改主库 my.cnf，主要是设置个不一样的 ID，以及要同步的数据库的名字。我们可以用 vim 编辑 /usr/local/webserver/mysql/my.cnf 文件，增加如下内容：

```
server-id = 1
log-bin = binlog
binlog-do-db = yuhongchun
```

yuhongchun 为要同步的数据库的名字。

2) 启动主库生效，命令如下：

```
nohup /usr/local/webserver/mysql/bin/mysqld_safe
--defaults-file=/usr/local/webserver/mysql/my.cnf &
```

3) 登录主库。

```
#/usr/local/webserver/bin/mysql -u root -p -S /tmp/mysql.sock
```

4) 赋予从库权限账号，允许用户在主库上读取日志，命令如下：

```
mysql> grant replication slave on *.* to 'admin'@'192.168.1.07' identified by '123456'
```

此操作完成以后，建议立即在另一台主机上验证一下，如果成功显示 mysql > 登录界面则表示设置成功，如下所示：

```
[root@localhost src]# mysql -u admin -p -h 192.168.1.106 -S /tmp/mysql.sock
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.5.3 - m3 - log Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql >
```

这里有个知识点要说明一下，我曾经也在这里犯过迷糊：MySQL 的权限系统在实现上比较简单，相关权限信息主要存储在几个被称之为 grant tables 的系统表中，即：mysql.user、mysql.db、mysql.host、mysql.table_priv 和 mysql.column_priv。由于权限信息的数据量比较小，访问又非常频繁，所以 MySQL 在启动的时候，就会将所有的权限信息都加载到内存中，并保存在几个特定的结构里。这就使得每次手动修改了相关权限表之后，都需要执行 flush privileges，通知 MySQL 重新加载 MySQL 的权限信息。当然，如果通过 grant、revoke 或 drop user 命令来修改相关权限，则不需要手动

执行 flush privileges 命令了。

5) 检查创建是否成功, 命令如下:

```
Mysql> select user, host from mysql.user;
+-----+-----+
| user | host |
+-----+-----+
| root | %    |
| root | 127.0.0.1 |
| admin | 192.168.1.107 |
| root | ::1  |
|      | localhost |
| root | localhost |
|      | mysql.cn7788.com |
| root | mysql.cn7788.com |
+-----+-----+
8 rows in set (0.00 sec)
```

6) 锁主库表。

```
mysql> flush tables with read lock;
```

7) 显示主库信息。

记录 File 和 Position, 从库设置将会用到:

```
mysql> show master status;
+-----+-----+-----+-----+
| File | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+-----+-----+-----+-----+
| binlog.000017 | 1187 | | |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

8) Slave 端机器获取 Master 端 MySQL 数据库的“快照”时有两种方法, 第一种是锁表后直接用 tar 将原 Master 打包给 Slave 机器, 这种情况适用于网站初始化时, 而且数据库比较单一。如果是已在线了一段时间, 而且机器上还存在着不同的数据库, 那么就不适合用 tar 打包了, 比如我数据库上面的生产数据库差不多有 9.8GB, 321 张表, 但这个数据库我不需要同步, 所以我们可以用别的方法来进行, 比如 mysqldump, 下面我会重点讲解这种方法。

如果只需单独备份主 Master 机器上的 yuhongchun 数据库, 我们可以用如下命令:

```
mysqldump --master-data -u root -p yuhongchun -S /tmp/mysql.sock > yuhongchun.sql
```

另外, 稍微解释一下 --master-data 参数的作用, mysqldump 程序的开发者给程序设计这项参数是为了帮助获取对应的 Log Position, 在添加了这个参数选项以后, mysqldump 会在 dump 文件中产生一条 CHANGE MASTER TO 命令, 命令中记录了 dump 时刻所对应的详细的 Log Position 信息。

2. 设置从库

1) 在主库上将 yuhongchun.sql 传输给从机, 我推荐用 rsync, 如果数据库比较大, 比如说几十 GB 的生产数据库, 那么用 rsync 尤其适合, 命令如下:

```
rsync -vzrtopg yuhongchun.sql root@192.168.1.107:/root/
```

2) 登录从库，建立 yuhongchun 数据库。

```
#mysql -u root -p -S /tmp/mysql.sock
mysql> create database yuhongchun;
```

然后，退出 mysql 命令行，导入 yuhongchun.sql。

```
#mysql -u root -p yuhongchun -S /tmp/mysql.sock < /root/yuhongchun.sql
```

3) 解锁主库表，命令如下：

```
mysql> unlock tables;
```

4) 修改从库 my.cnf，命令如下：

```
# vim /usr/local/webserver/mysql/my.cnf
#slave
server-id=2
#master-host=192.168.1.106
#master-user=admin
#master-password=123456
```

其实上面的最后三行代码不需要添加，只需要到 server-id=2 即可。

5) 在从库上设置同步。

设置连接 MASTER MASTER_LOG_FILE 为主库的 File，MASTER_LOG_POS 为主库的 Position，命令如下：

```
mysql> slave stop;
mysql> change master to master_host='192.168.1.106',master_user='admin',master_password='123456',
master_log_file='binlog.000017', master_log_pos=1187;
mysql> slave start;
```

6) 查看从库的 status 状态，命令如下：

```
mysql> show slave status\G;
```

显示状态界面如图 6-11 所示。

注意图中方框中的两部分内容：Slave_IO_Running: Yes (网络正常)；Slave_SQL_Running: Yes (表结构正常)，这两部分必须都为 Yes 才行。

7) 进行测试。

在主库上的 yuhongchun 表上建立名为 yuhongchun 的表，如下所示：

```
mysql> CREATE TABLE 'yuhongchun' (
'id' INT(5) UNSIGNED NOT NULL AUTO_INCREMENT,
'username' VARCHAR(20) NOT NULL,
'password' CHAR(32) NOT NULL,
'time' DATETIME NOT NULL,
'number' FLOAT(10) NOT NULL,
'content' TEXT NOT NULL,
PRIMARY KEY ('id')
```



```
) ENGINE = MYISAM;
```

```

      Master_Port: 3306
      Connect_Retry: 60
      Master_Log_File: binlog.000017
      Read_Master_Log_Pos: 1187
      Relay_Log_File: relaylog.000002
      Relay_Log_Pos: 250
      Relay_Master_Log_File: binlog.000017
      Slave_IO_Running: Yes
      Slave_SQL_Running: Yes
      Replicate_Do_DB:
      Replicate_Ignore_DB: mysql,test,information_schema
      Replicate_Do_Table:
      Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
      Replicate_Wild_Ignore_Table:
      Last_Errno: 0
      Last_Error:
      Skip_Counter: 0
      Exec_Master_Log_Pos: 1187
      Relay_Log_Space: 399
      Until_Condition: None
      Until_Log_File:
      Until_Log_Pos: 0

```

图 6-11 从 MySQL 的 status 状态显示

在从表中马上可以看到效果，主从同步成功了。我们接着可以用工具在 yuhongchun 的表上增加记录。在 Windows XP 下图形化操作 MySQL 数据库的工具还是有很多的，有的人喜欢用 phpMyadmin，我则喜欢用 SQLyog Enterprise 专业版，它的好处是可以直接在 Windows XP 下进行图形操作，而不需要考虑服务器是不是 PHP 环境。试想一下，如果线上环境都是 JSP 开发环境，那应该如何部署 phpMyadmin 呢（它是基于 PHP 开发的）？下面我们可以看一下利用 SQLyog Enterprise 专业版往数据表里增加两条记录的界面图，如图 6-12 所示。



The screenshot shows the SQLyog Enterprise Professional Edition interface. At the top, there are tabs for '1 结果', '2 Profiler', '3 信息', '4 表数据', '5 Info', and '6 历史'. Below the tabs, there are controls for 'All Row', 'Rows in a Range', 'First Row', 'No. of Rows' (set to 50), and a 'Refresh' button. The main area displays a table with the following data:

id	username	password	time	number	content	
<input type="checkbox"/>	2 andrewyu	7788	1980-08-10 01:00:00	7788	wo shi xue	10B
<input checked="" type="checkbox"/>	1 yuhongchun	5566	1980-01-01 00:00:00	5566	hello, world	11B
*	(NULL) (NULL)	(NULL)	(NULL)	(NULL) (NULL)		OK

图 6-12 SQLyog Enterprise 专业版工作图

然后，我们可以在 Slave 机上查看相关情况如下：

```

mysql> select * from yuhognchun;
+----+-----+-----+-----+-----+-----+
| id | username | password | time | number | content |
+----+-----+-----+-----+-----+-----+
| 1 | yuhongchun | 5566 | 1980-01-01 00:00:00 | 7788 | hello world |
| 2 | andrewyu | 7788 | 1980-08-10 00:00:00 | 7788 | hello world |
+----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

```

整个配置过程非常顺利，一次到位，感觉 MySQL 的 Replication 用起来还是很简单方便的。

MySQL 主从 Replication 复制非常快，小数据量的改变几乎感觉不到延迟（但还是属于异步 Replication），通常在 Master 端改动以后，Slave 端也会立即改动，这种模式非常适合那种对延时性要求很低的工作环境，比如 BBS 论坛。不过，用 MySQL 的 Replication 做备份时就会发现它的弊端，如果 Master 端有误操作的话，Slave 也会收到命令，以讹传讹，这会非常麻烦。所以，如果是作为备份机使用的话，我们就应该采取延时 Replication 的方法，通常是延迟一天，这不是本章节重点，大家可以自行研究。

6.7.9 可扩展性设计之数据切分

通过 MySQL Replication 功能所实现的扩展总会受到数据库大小的限制，一旦数据库过于庞大，尤其是当写入过于频繁，很难由一台主机支撑时，我们会面临扩展瓶颈。那么也就必须找其他技术手段来解决这个瓶颈了。用什么方法来解决呢？这就是我要向大家介绍的数据切分。

数据的切分（又叫 Sharding）根据切分规则的类型，可以分为两种切分模式。一种是按照不同的表或 schema 切分到不同的数据库或主机上，这种切分模式可以称之为数据的垂直切分；另外一种则是根据表中数据的逻辑关系，将同一个表中的数据按照某种条件拆分到多台数据库或主机上，这种切分模式称之为数据库的水平切分。

垂直切分的最大优点就是规则简单，实施起来也很方便，尤其适合各业务之间的耦合度非常低、相互影响很小、业务唤回非常清晰的系统。在这种系统中，可以很容易地做到将不同业务模块所使用的表拆分到不同的数据库中。根据不同的表来进行拆分，对应用程序的影响也很小，拆分规则也会比较简单清晰。

水分切分与垂直切分相比，稍微复杂了一些。因为要将同一个表中的不同数据拆分到不同的数据库中，对于应用程序来说，拆分规则本身就比根据表名拆分更为复杂，后期的数据维护也会更复杂。

当某些表的数据量和访问量特别大时，可能通过垂直切分将其放到独立的设备上后仍然无法满足性能要求，这时就必须将垂直切分和水平切分相结合了，先垂直切分，然后再水平切分，这样才能解决这种超大型表的性能问题。

在实际工作环境中，这 3 种切分方式都会用到，不过，它们都有哪些优缺点呢？下面我们一起来看看。

垂直切分的优点：

- 数据库的拆分简单明了，拆分规则明确。
- 应用程序模块清晰明确，整合容易。
- 数据维护方便易行，容易定位。

垂直切分的缺点：

- 部分表关联无法在数据库级别完成，要在程序中完成。
- 对于访问极其频繁且数据量超大的表仍然存在性能瓶颈，不一定能满足要求。
- 事务处理相对复杂。
- 切分达到一定程度以后，扩展性会受到限制。
- 过度切分可能会导致系统过于复杂而难以维护。

水平切分的优点：

- 表关联基本能够在数据库端完成。
- 对于某些数据量超大且高负载的表来说，不会遇到性能瓶颈。
- 应用程序端的整体架构改动相对较少。
- 事务处理相对简单。
- 只要切分规则能够定义好，基本上较难遇到扩展性限制。

水平切分的缺点：

- 切分规则相对复杂，很难抽象出一个能够满足各个数据库的切分规则。
- 后期数据的维护难度有所增加，人为手动定位数据更困难。
- 应用系统各模块耦合度较高，可能会对后面数据的迁移拆分造成一定的困难。

联合切分的优点：

- 可以充分利用垂直切分和水平切分各自的优势而避掉了各自的缺陷。
- 让系统扩展性得到了最大化提升。

联合切分的缺点：

- 数据库系统架构比较复杂，维护难度更大。
- 应用程序架构也更为复杂。

我们通过数据切分技术将一个大的 MySQL Server 切分成多个小的 MySQL Server，既解决了写入性能瓶颈问题，同时也再一次提升了整个数据库集群的扩展性。不论是垂直切分还是水平切分，都能够让系统遇到瓶颈的可能性更小，这种技术尤其适合那些并发量大的游戏领域。

6.8 生产环境下的 MySQL 数据库备份

生产环境因为其特殊性（既不能重启也不能关机），有别于我们平时学习测试的实验环境的开发环境，所以不能将 MySQL 数据库停下来做全备。在企业中运行 MySQL 数据库就是这样的生产环境，它必须 365 × 24 小时不间断地为我们服务，所以这时候我们必须要考虑很多相关因素。其最典型的特点是，生产环境下是绝对不允许宕机的！怎么可能仅仅为了备份而宕机呢？想都不要想啦！我们要清楚备份只是为了保障我们的系统更加安全地运行，减少灾难带来的损失，相对而言它只是一个小角色，设想一下，如果服务器压根就没有发生问题的可能性，那么备份也就没有存在的必要了，只是因为可能会有意外发生，所以我们才要采取相应的措施。我们也可以看看服务器都会发生哪些意外，有下面几种情况：

- 人为操作失误造成某些数据被误操作。
- 硬件故障造成数据库部分或全部数据丢失。
- 因为存在安全漏洞，入侵者将数据恶意破坏。

看起来数据库的备份是一个必须完成的工作，而且我们必须针对于各自不同的生产环境，制订出完美的备份方案来。

MySQL 的备份主要分为逻辑备份和物理备份。在备份之初我们需要考虑哪些因素呢？

- 首先确定当前 MySQL 是处在哪种表类型下工作的，它们支持事务处理还是非事务的，因为我们需要根据不同的特点来做一些设置。
- 其次要选择备份的形式是全备份还是增量备份，它们各有优缺点。
- 为了确保恢复的完整性，我们得开启 binary log 功能，同时 binlog 给恢复工作也带来了很大

的灵活性，可以基于时间点或是位置进行恢复。考虑到数据库性能，我们可以将 binlog 文件保存到其他安全的硬盘中。

- 正如最初所提到的，备份操作和应用服务得同时运行，这样就十分消耗系统资源了，会导致数据库服务性能下降，这就要求我们选择一个合适的时间（比如在应用负担很小的时候），再来进行备份操作。
- 最后要提的就是，不是备份完就万事大吉，我们还得确认备份是否可用，所以之后的恢复测试是完全有必要的。

下面就来具体实践一下。

1. 逻辑备份和恢复

MySQL 的逻辑备份就是将数据库的结构连同数据用一个文本文件备份出来，我们可以查看并编辑这个文件。逻辑备份对于所有的存储引擎来说都是适用的，而后面提到的物理备份则会根据各数据库不同的特点采取不同的备份方法。MySQL 逻辑备份的主要工具就是大家熟悉的 MySQL 自带的 `mysqldump` 工具，其命令如下：

```
mysqldump -u root -p -all -database > all.sql
```

下面的命令是备份所有数据库：

```
mysqldump -u root -p mysqlsystems > mysqlsystems.sql
```

下面的命令是备份指定数据库 `mysqlsystems`：

```
mysqldump -u root -p mysqlsystems wp discuz > wp_discuz.sql
```

上面的命令会备份指定数据库中的若干表，例如 `mysqlsystems` 中的表 `wp discuz`。

说明 MyISAM 中为了保持数据的一致性，需要在备份之前对所备份的数据库加读锁操作 `flush table with read lock`；InnoDB 则可以在 `mysqldump` 命令中加入 `-single-transaction` 选项，生成一个快照来保证数据备份期间的一致性。

以上介绍的都是全备份，我们也知道全备份是必要的，虽然在应用之初我们可以采用全备份的方式，但随着应用数据的增加，这种备份方式的效率就会变得很低，每次备份都会花掉大量的时间和系统资源。我们有必要在全备份的同时结合使用增量备份。增量备份就是通过备份 binlog 日志来实现的，当我们开启 MySQL 的二进制日志功能时，对数据库进行的 DML（select 除外）、DDL 操作都会记录到其中（MySQL Replication 也是通过 binlog 日志来实现的）。每当 MySQL 重启时，数据库就会重新生成一个 binlog 文件。比如 5 月 9 日凌晨 2 点开始备份，同时我们也会将 binlog 进行更新并重新生成一个，那么备份完成之后，我们对数据库进行的操作就会记录到新生成的这个二进制文件中，增量备份就是要我们保存从 5 月 9 日 2 点开始，到我们进行下一次备份工作之间的备份，这期间的操作都在这个 binlog 文件里面，我们只需将它备份就可以了。

恢复则比较简单了，命令如下：

```
mysql -u root -p mysqlsystems < mysqlsystem.sql
```

上面的命令恢复了全备份的信息。


```
mysqlbinlog binlog -file | mysql -u root -p
```

上面的命令恢复了增量备份的部分信息。

2. 物理备份和恢复

物理备份分为冷备份和热备份。物理备份其实就是物理文件的复制，从一个存放位置拷贝到另一个位置。

冷备份一般很少使用，原因我们都知道，应用是坚决不允许停机的，即便时间很短也不行，况且备份的时间和文件大小成正比，而且文件只会越来越多，不会减少。

这里主要介绍热备份，根据存储引擎的不同将采用不同的备份工具。

□ MyISAM

方法一：MySQL 提供了一个实用程序 `mysqlhotcopy`，这个程序是用 Perl 编写的，其主要实现原理是先锁表，然后执行 `flush tables` 动作，该正常关闭的表正常关闭，该进行 `flush` 同步的数据都进行同步，然后通过复制命令，将需要备份的表或数据库的所有物理文件都复制到指定的备份位置上。它的语法也很简单，命令如下：

```
mysqlhotcopy 数据库 备份目录
```

方法二：在 SQL 命令行下执行 `flush tables for read`，将数据库中所有的表加读锁，以保证数据的一致性，然后通过 `cp` 命令将数据文件备份到指定目录下。MyISAM 是 MySQL 的默认表类型，它是基于 ISAM 代码的，但有许多有用的扩展。

□ InnoDB

对于 InnoDB，可以使用 percona 公司的 `xtrabackup`，或是 Innobase 自己的 `ibbackup`（收费产品）。

我们经常用的方法是：

1) 通常情况下，我们的 MySQL 至少是一主一从，很多时候我们可以通过暂时停止 Slave 上面的 SQL 线程来让 Slave 机器停止所有数据库的写入操作，然后就可以在线进行备份操作了。

2) 对于较为核心的在线应用数据库，尽量让备机的数据延后主机的时间，在一定的时间段之内即可，一般来说延后一天是一个比较常规的做法。

3) 对于一些搭建的临时数据库的备份，仅仅通过一个逻辑全备份即可满足需求，即不需要使用二进制日志来进行恢复。

4) 为了防止本地备份硬盘因为频繁地读写数据而发生损坏，我们备份数据时都采用本地 + 异地双备份的方案，具体脚本文件请大家参考第 5 章的内容。

在进行备份的过程中，值得注意的问题如下：

- 确定要备份的表的存储引擎是事务型还是非事务型，两种不同的存储引擎在备份时会以不同的方式处理数据的一致性问题。
- 确定使用全备份还是增量备份。全备份的优点是可保持最新备份，恢复的时候花费的时间相对较少；缺点是如果数据量大，将会花费很多的时间来进行备份，并且会给系统造成时间上的压力。增量备份则恰恰相反，只需要备份每天的增量日志，备份时间少，对负载压力也小；缺点就是恢复的时候需要使用全备份加上次备份到故障前的所有日志，恢复时间会长些。
- 可以考虑采取复制的方法来做异地备份，但是要记住，复制不能代替备份，它对数据库的误操作也无能为力。

- 要定期做备份，备份的周期要充分考虑系统可以承受的恢复时间。备份要在系统负载较小、网络流量较小的时候进行（不能因为备份工作而影响正常运行）。
- 确保 MySQL 打开了 log-bin 选项，有了 binlog，MySQL 才可以在必要的时候做完整恢复，或者基于时间点的恢复，或者基于位置的恢复。另外，在 my.cnf 里有一个非常有用的设置：expire_logs_days = 90，它的作用是设置二进制日志过期自动移出的时间限制，若到达设定的最大保留期限，MySQL 会自动删除过期的二进制日志。它的最大值是 99，默认为 0（即永不过期），我们的做法是 expire_log_days = 0，然后手动清理备份二进制日志，这样出错的几率会更低。
- 由于我们配置的是 expire_log_days = 0，即永不过期，所以清理 MySQL 日志都是手动完成的，这样也就带来了一个问题，MySQL 服务器的硬盘空间常常会爆满，搞得 Nagios 狂发报警信息，这一点我们在工作中也要注意。
- 要经常做备份恢复测试，确保备份是有效的，并且是可以恢复的。

相信大家看完此节后，会对生产环境下 MySQL 的备份方式有一个大致的了解，我们也可以针对自己的生产环境，对 MySQL 数据库进行备份。俗话说得好，备份是救命的稻草，所以我们应该将其提升到自己的工作议程中，并引起足够的重视。

6.9 部分项目施工图纸

部分项目施工图纸如图 6-13、图 6-14 和图 6-15 所示。

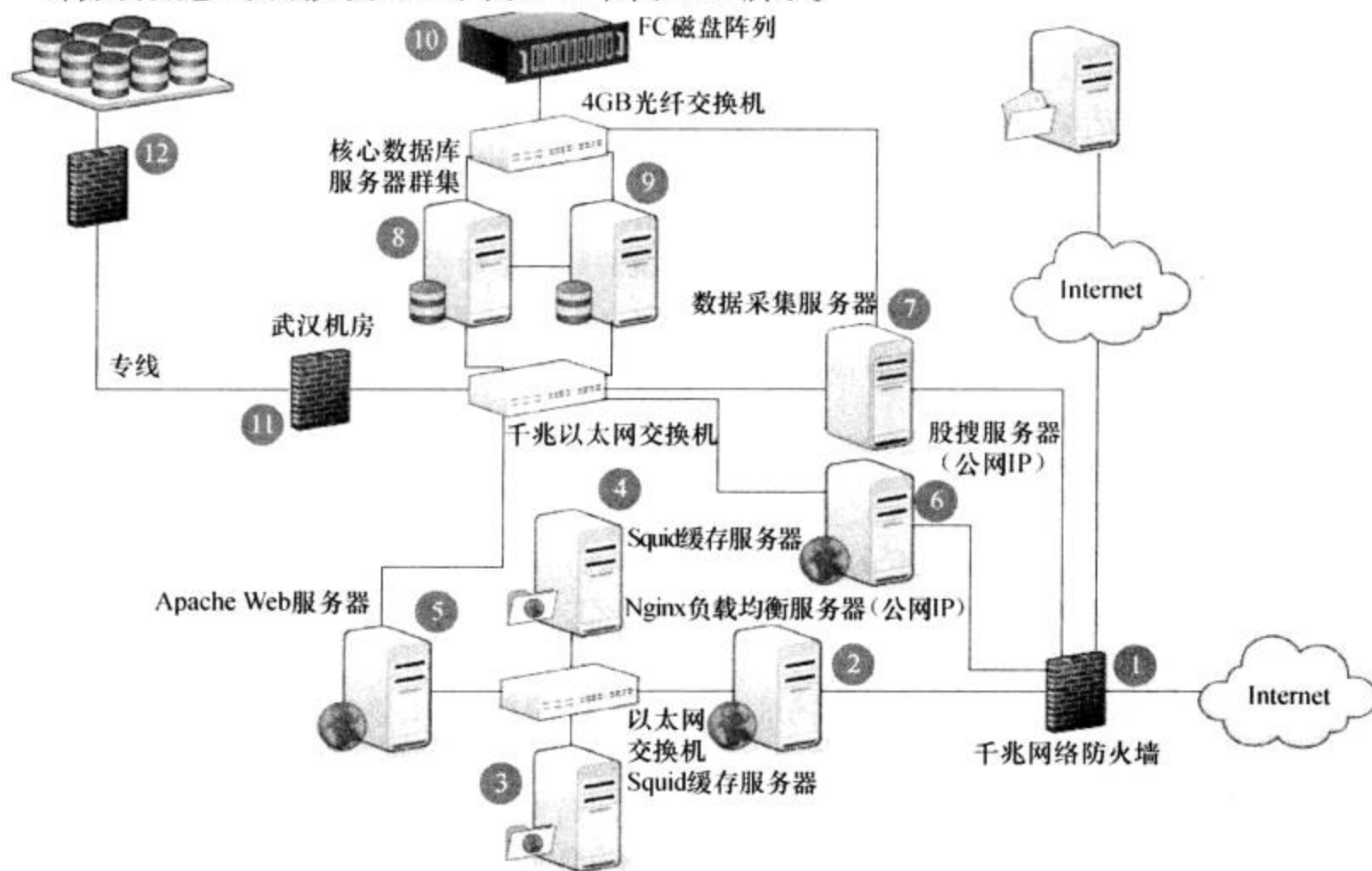


图 6-13 Linux 集群项目实施图纸之一

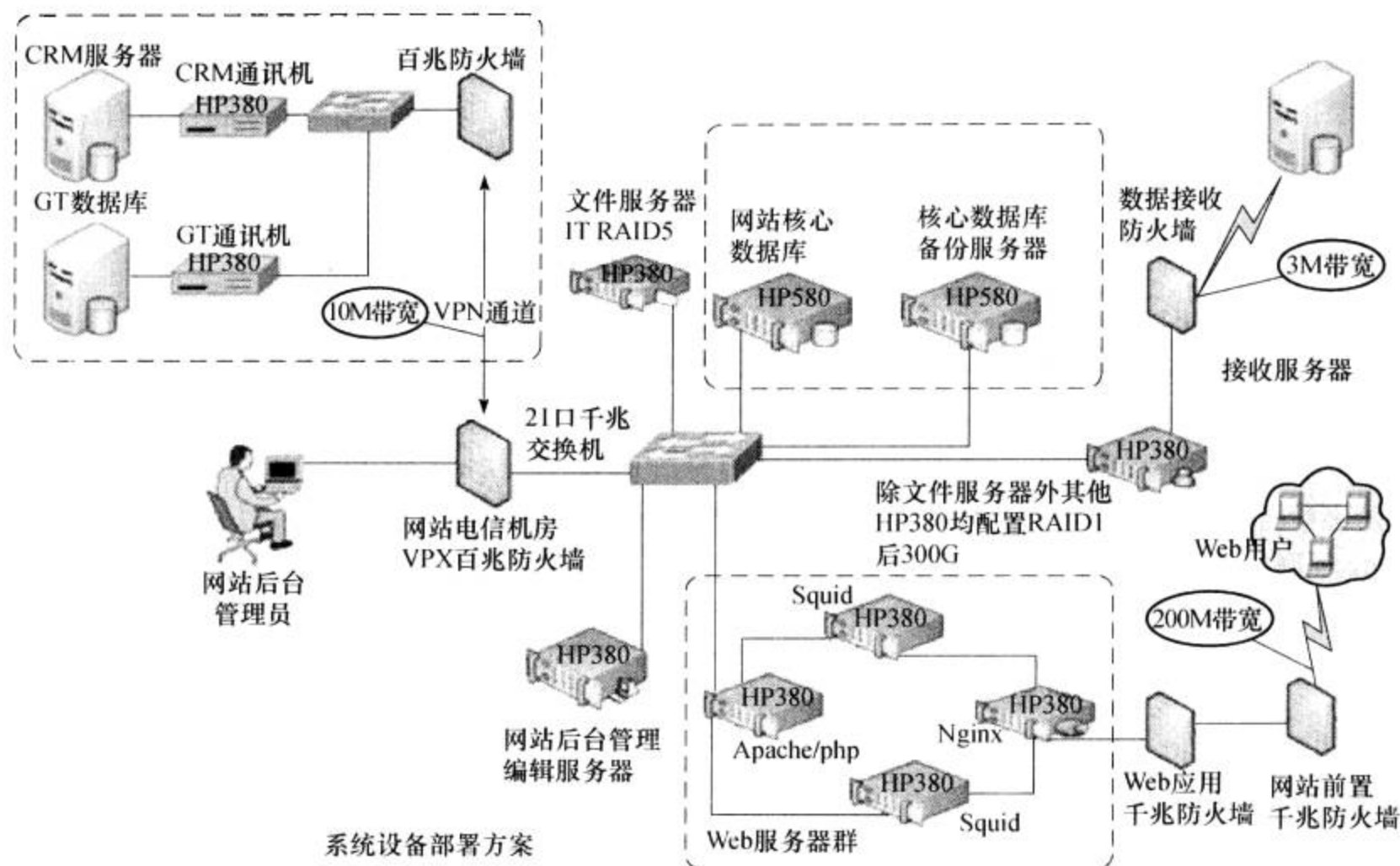


图 6-14 Linux 集群项目实施图纸之二

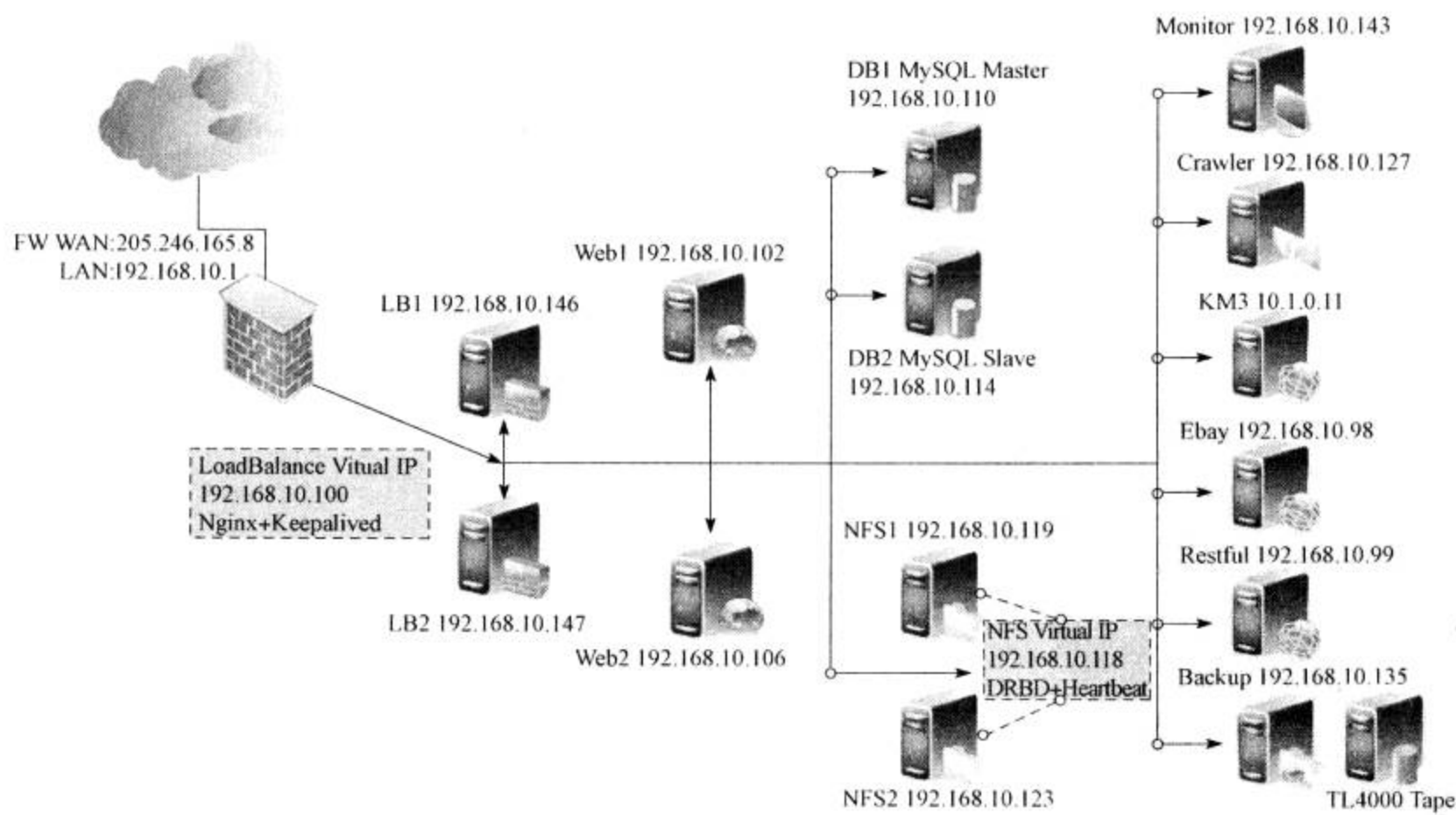


图 6-15 Linux 集群项目实施图纸之三

为了保护公司及客户隐私，我在项目施工图纸上做了无害处理，如图 6-14 和图 6-15 所示。在项目施工前期均是以单 Nginx 来作负载均衡器的，测试阶段则用的是 Nginx + Keepalived 负载均衡高可用架构，防火墙用的是华赛防火墙 USG5000，防火墙的工作模式采用的是路由模式（没有采用透明模式），仅仅映射内网的 80 端口和 443 端口到公网 IP 上去（即 DNAT 模式）。

6.10 小结

这一章主要向大家介绍了 Linux 集群技术所采用的软件，例如 F5/LVS、HAProxy 及 Nginx，还有 DRBD + Heartbeat 等。另外我还通过真实项目着重介绍了现在比较流行的 LVS + Keepalived、Nginx + Keepalived 这两种负载均衡高可用技术，它们相对比较成熟。接着，我又向大家介绍了 MySQL 数据库的单机调优及集群架构设计，包括 MySQL Replication 及 MySQL Cluster 等。相信通过阅读这章内容，大家会对生产环境下的 Linux 集群有所了解。现在越来越多的公司和企业意识到了 Linux 集群的稳定和高效，所以都考虑采用它为企业提供负载均衡高可用方案。我希望大家能熟练掌握 Linux 集群相关知识，为自己的职业技能添加技术含金量，这对自身的成长也是非常有帮助的。



第 7 章 VPN 在企业中的部署应用

- 7.1 流行的 VPN 技术及其分类
- 7.2 如何选择自己需要的 VPN
- 7.3 IPSec VPN 的不足
- 7.4 OpenVPN 的应用范畴
- 7.5 经典企业 VPN 部署案例
- 7.6 部署 OpenVPN 服务器的注意事项
- 7.7 小结

VPN 的英文全称是“Virtual Private Network”，译为“虚拟专用网络”。VPN 被定义为通过一个公用网络（通常是 Internet）建立一个临时的、安全的连接，是一条安全、稳定的隧道。使用这条隧道可以对数据进行加密，达到安全使用互联网的目的。虚拟专用网是对企业内部网的扩展。虚拟专用网可以帮助远程用户、公司分支机构、商业伙伴及供应商与公司的内部网建立可信的安全连接，它是经济有效地连接到商业伙伴和用户的安全外联网的虚拟专用网。VPN 主要采用隧道技术、加解密技术、密钥管理技术和使用者与设备身份认证技术。

VPN 的命名基于以下 3 个因素：首先，它是虚拟（virtual）的，它使用一个已经存在的基础设施；其次它是专用（private）的，通过一个安全的协议来封装数据；最后它是一个网络（network），它将两个或多个设备（或者网络）连接起来。

VPN 可以提供的功能主要有：防火墙功能、认证、加密、隧道化。

VPN 可以通过特殊加密的通信协议连接到 Internet 上，在位于不同地方的两个或多个企业内部网之间建立一条专有的通信线路，就好比是架设了一条专线一样，但是它并不需要去铺设真正的光缆之类的物理线路。这就好比去电信局申请专线，但是不用付铺设线路的费用，也不用购买路由器等硬件设备。VPN 技术原是路由器上具有的重要技术之一，在交换机、防火墙设备或 Windows Server 2003 及 FreeBSD 等操作系统上，当然还有 Linux 系统中都支持 VPN 功能。我们可以这样简单地理解 VPN 技术，VPN 的核心技术就是利用公共网络建立虚拟专用网，它不是真的专用网络，但是却能实现专用网络的功能。VPN 如今已是非常流行的技术，对于远程工作者及短期连接而言，使用 VPN 比租用专用的连接线路更具有实用价值。

7.1 流行的 VPN 技术及其分类

从总体上来说，VPN 技术非常复杂，它涉及通信技术、密码技术和现代认证技术，是一门交叉科学。目前，VPN 主要包含隧道技术与安全技术。下面，我就企业和市面上应用得多的 VPN 技术做一下归纳和总结。

1. PPTP

点对点隧道协议（PPTP）是由包括微软和 3Com 等公司组成的 PPTP 论坛开发的一种点对点隧道协议，基于拨号使用的 PPP 协议使用 PAP 或 CHAP 之类的加密算法，或者使用 Microsoft 的点对点加密算法 MPPE。其通过跨越基于 TCP/IP 的数据网络创建 VPN 实现了从远程客户端到专用企业服务器之间的数据安全传输。PPTP 支持通过公共网络（例如 Internet）建立按需的、多协议的、虚拟专用网络。PPTP 允许加密 IP 通信，然后在要跨越公司 IP 网络或公共 IP 网络（如 Internet）发送的 IP 头中对其进行封装。

2. L2TP

第 2 层隧道协议（L2TP）是 IETF 基于 L2F（Cisco 的第二层转发协议）开发的 PPTP 后续版本，是一种工业标准的 Internet 隧道协议，可以为跨越面向数据包的媒体发送点到点的协议（PPP）框架提供封装。PPTP 和 L2TP 都使用 PPP 协议对数据进行封装，然后添加附加包头用于数据在互联网上的传输。PPTP 只能在两端点间建立单一隧道，L2TP 则支持在两端点间使用多隧道，用户可以针对不同的服务质量创建不同的隧道。L2TP 可以提供隧道验证，而 PPTP 则不支持隧道验证。但是当 L2TP 或 PPTP 与 IPSEC 一起共同使用时，可以由 IPSes 提供隧道验证，不需要在第 2 层协议上使用 L2TP 验证隧道。PPTP 要求互联网络为 IP 网络，L2TP 只要求隧道媒介提供面向数据包的点对点

的连接, L2TP 可以在 IP (使用 UDP)、帧中继永久虚拟电路 (PVCs)、X.25 虚拟电路 (VCs) 或 ATM VCs 网络上使用。

3. IPSec 隧道模式

隧道是封装、路由与解封装的整个过程。隧道将原始数据包隐藏 (或封装) 在新的数据包内部。该新的数据包可能会有新的寻址与路由信息, 从而能够通过网络传输。隧道与数据保密性结合使用时, 在网络上窃听通信的人将无法获取原始数据包数据 (以及原始的源和目标)。封装的数据包到达目的地后, 会删除封装, 原始数据包头用于将数据包路由到最终的目的地。

隧道本身是封装数据经过的逻辑数据路径, 对原始的源和目的端, 隧道是不可见的, 而只能看到网络路径中的点对点连接。连接双方并不关心隧道起点和终点之间的任何路由器、交换机、代理服务器或其他安全网关。将隧道和数据保密性结合使用时, 可用于提供 VPN。

封装的数据包在网络中的隧道内部传输。在上面的例子中, 该网络是 Internet。网关可以是外部 Internet 与专用网络间的周界网关。周界网关可以是路由器、防火墙、代理服务器或其他安全网关。另外, 在专用网络内部可使用两个网关来保护网络中不信任的通信。

当以隧道模式使用 IPSec 时, 其只为 IP 通信提供封装。使用 IPSec 隧道模式主要是为了方便与其他不支持 IPSec 上的 L2TP 或 PPTP VPN 隧道技术的路由器、网关或终端系统之间进行相互操作。

4. OpenVPN

OpenVPN 允许参与建立 VPN 的单点使用预设的私钥、第三方证书, 或者用户名/密码来进行身份验证。它大量使用了 OpenSSL 加密库, 以及 SSLv3/TLSv1 协议。OpenVPN 能在 Linux、xBSD、Mac OS X 与 Windows 2000/XP 上运行。它并不是一个基于 Web 的 VPN 软件, 也不与 IPSec 及其他 VPN 软件包兼容。

OpenVPN 提供了多种身份验证方式, 用以确认参与连接双方的身份, 包括: 预享私钥, 第三方证书及用户名/密码组合。预享密钥最为简单, 但同时它只能用于建立点对点的 VPN; 基于 PKI 的第三方证书提供了最完善的功能, 但是需要额外的精力去维护一个 PKI 证书体系。OpenVPN 2.0 引入了用户名/口令组合的身份验证方式, 它可以省略客户端证书, 但是仍有一份服务器证书需要被用作加密。

OpenVPN 所有的通信都基于一个单一的 IP 端口, 默认且推荐使用 UDP 协议通信, 同时也支持 TCP。OpenVPN 连接能通过大多数的代理服务器, 并且能够在 NAT 的环境中很好地工作。服务端具有向客户端“推送”某些网络配置信息的功能, 这些信息包括: IP 地址、路由设置等。OpenVPN 提供了两种虚拟网络接口: 通用 Tun/Tap 驱动, 通过它们, 可以建立 3 层 IP 隧道; 虚拟二层以太网, 它可以传送任何类型的二层以太网数据, 传送的数据可通过 LZO 算法压缩。IANA (Internet Assigned Numbers Authority) 指定给 OpenVPN 的官方端口为 1194。OpenVPN 2.0 以后版本每个进程可以同时管理数个并发的隧道。

OpenVPN 使用通用网络协议 (TCP 与 UDP) 的特点使它成为 IPSec 等协议的理想替代品, 尤其是在 ISP (Internet service provider) 过滤某些特定 VPN 协议的情况下。在选择协议的时候, 需要注意两个加密隧道之间的网络状况, 如有高延迟或丢包较多的情况, 请选择 TCP 协议作为底层协议, UDP 协议由于存在无连接和重传机制, 导致隧道上层协议的数据重传, 其效率非常低下。

OpenVPN 使用 OpenSSL 库加密数据与控制信息。它使用了 OpenSSL 的加密及验证功能, 这也就意味着, 它能够使用任何 OpenSSL 支持的算法。它提供了可选的数据包 HMAC 功能, 以提高连接的安全性。此外, OpenSSL 的硬件加速也能提高它的性能。

OpenVPN 具备了许多与生俱来的安全特性：它在用户空间运行，无需对内核及网络协议栈做修改；初始完毕后以 chroot 方式运行，放弃 root 权限；使用 mlockall 以防止敏感数据交换到磁盘。

参考文档：http://www.360doc.com/content/09/1126/12/25127_9782087.shtml

7.2 如何选择自己需要的 VPN

某企业的规模越来越大了，在全国各地都有自己的分公司和门市店（大约有 100 多家），有时候需要将全部的资料共享，这时候选择一款 VPN 是最节约成本的方法，面对市面上如此之多的流行的 VPN 技术，我们该从哪些方面来选择 VPN 呢？我们可以从以下几方面来考虑：

1) 总部与远程用户的上网地点是否固定？一般来说，总部机房与远程分公司的机房是固定的，这种情况下可以考虑用 IPSec。IPSec VPN 非常适合固定机构的连网的网络环境，而需要灵活行动的用户则不适用这种 VPN。但远程分公司应该还有不少办公人员，他们的上网环境可能是不固定的，这个时候如果他们也想通过 VPN 连进公司的网络，分享公司的销售或其他资源，那么 PPTPD 和 OpenVPN 应在考虑范围之内。

2) 是否需要额外的硬件投入？如果这一项为 NO 的话，那么我们首先就要排除掉 IPSec，若采用 IPSec VPN，每一个连锁店都需要安装一台 VPN 硬件设备，这就意味着要多增加 100 多台 VPN 设备（分公司和门市店大约有 100 多家），因此成本将大大增加，不能达到公司的预算要求。而其他几种 VPN，包括 PPTPD/L2TP、OpenVPN 都是开源软件，不需要额外投入硬件，这些可以在我们的考虑范围之内。

3) 远程连接的管理是否方便？IPSec 客户端的配置是很烦琐的，相对而言，PPTPD 和 OpenVPN 就简单容易多了。

4) 是否需要单独为 VPN 划分网络？这一点属于网络规划和安全的问题，按照正常做法，我们应该单独为 VPN 环境划分网络，这也是保证企业敏感资料安全性的方法。

我们在为企业挑选 VPN 时，可以尽量从以上 4 点来考虑，成本问题、安全性、易用性更是考虑的首要条件。

7.3 IPSec VPN 的不足

IPSec VPN 是几年前相当流行的 VPN 技术，但是它自身也有一些问题，在实际投入时会发现它的不足之处，这里我可以举个例子来说明一下。

小王是某连锁服装公司的网管。公司总部位于广州，在全国各大城市开设了 100 多家连锁销售店，每家店都拥有计算机 10 台左右。2008 年，为了实施信息化管理与经营的策略，企业部署了 IPSec VPN，将广州总部与各个连锁店的网络连接起来，实现了物流、财务、产品等信息共享。今年，公司计划新开 100 家连锁店，但是高层领导授意小王，根据企业财务方面的现状，采购 VPN 设备的预算必须得到控制。那么，如果小王依然采用 IPSec VPN 的方案，就会出现以下 3 个问题：

1) 成本大增。若采用 IPSec VPN，每一个连锁店都需要安装一台 VPN 硬件设备，这就意味着要多增加 100 台 VPN 设备，因此成本将大大增加，不能达到公司的预算要求。

2) 管理与维护的复杂度增加。对于新加入的外点远程用户，IPSec VPN 的配置非常复杂，一个客户端的参数配置可达 20 多个步骤，得花近 10 分钟的时间。同时，加上合作伙伴、高管等移动 VPN 的使用需求，将大大增加网管的工作量。

3) 信息的安全性无法保障。虽然 IPSec VPN 的整体安全性相当高，但是，它无法划分内网使

用者的访问权限。就是说，一旦与总部联机成功，任何一个用户都能够访问整个企业网络。如果这个用户碰巧是一个不怀好意的员工或商业间谍，这种情况就很危险了。

以上的3个方面是当前 IPSec VPN 企业用户遇到的普遍问题。而这些都是 OpenVPN 的优势所在，它属于开源免费的软件，采用了 OpenSSL 安全认证，客户端配置相当容易。

7.4 OpenVPN 的应用范畴

目前 OpenVPN 应用得比较多，我跟几个做系统和网络的朋友专门聊过这个话题，并总结了其应用范畴如下：

1) 使用 OpenVPN 实现网通、电信机房间快速安全的通信。具体实现方式也不复杂，如果维护的网站是属于 CDN 系统的话，SA 们都会选择这种方式来连接电信和网通的机房，以便快速同步数据。比方说，我们现在有3个机房，一个是本地的双线机房，一个是电信的，一个是网通的，我们可以拿双线机房的某台机器作为 OpenVPN Server，另外电信和网通的各拿一台机器作为 OpenVPN Client，这样3台机器之间就形成了一个 VPN 网络，相互之间都可以传输数据，而且在传输数据时不仅快速而且安全。

2) 使用 OpenVPN 作为公司的远程拨入服务器，OpenVPN 可以选择网桥和路由模式，网络环境简单的可以用路由模式，复杂的推荐用网桥模式，网桥模式尤其适合那种网段多、路由复杂的办公环境。

3) OpenVPN 尤其适合作为局对局的 VPN 服务器，比方说小区上网环境的客户端可实现局对局环境的 VPN，其他 VPN 很多都不能满足这种需求。

4) OpenVPN 可以穿透一些顽固型的防火墙，直接访问其后的局域网环境中的资源（这点就比较强悍，具体作用要看我们在企业中的需求），穿透性强算是 OpenVPN 的隐含特性，在实际环境中我们发现此功能确实强大。

7.5 经典企业 VPN 部署案例

7.5.1 案例一：在 Centos5.5 x86_64 下单网卡配置 PPTPD 服务器

由于是办公环境，所以整个网络环境架构得比较简单，如图 7-1 所示。

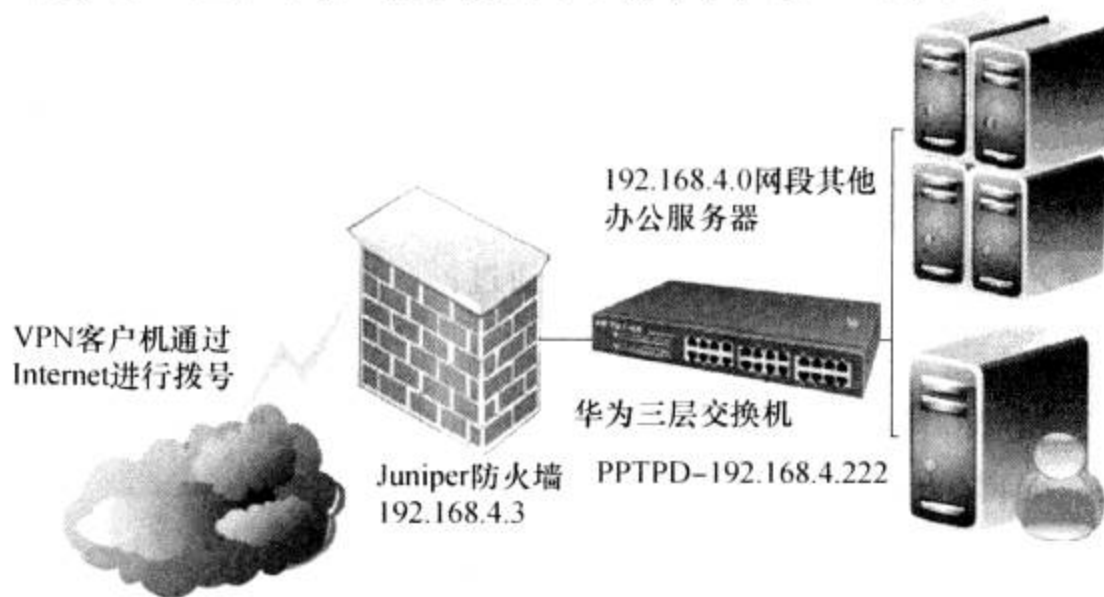


图 7-1 办公环境下的网络拓扑图

这里解释一下公司的拓扑和需求：

公司的办公网络是 192.168.4.0/24，均只用单网卡 eth0，通过 Juniper 防火墙映射公网 IP 上网，即内网内所有机器的网关均是防火墙的固定 IP 192.168.4.3，目前希望通过外网拨号（VPN 服务器 IP 为 192.168.4.222）可进入公司内部局域网办公，因为内网里还有 SVN、Samba 文件服务器及 FTP 资料下载库等，即要将 VPN 办公环境要做成点对局的。

初期因为我有成功的 OpenVPN 点对点案例，所以考虑用 OpenVPN 来做，不料却失败了。点对点很顺利，但点对局却失败了。大家可看一下 OpenVPN 的关键之处。

关键一：取消注释 `push "route 192.168.4.0 255.255.255.0"`，并将地址和掩码替换为办公网络的地址和掩码，目的是为客户端加一条路由，这样客户端才有可能访问到办公网络中除 VPN Server 之外的其他主机（有很多 VPN 客户端是直接添加默认路由的，这样客户端的所有连接请求都被路由到 VPN 通道内，这就会导致客户端此时不能访问 VPN，而添加了指定地址的路由就不会出这一问题）。

关键二：Server 端开启 IP 转发。

配置服务器，运行包转发：`echo "1" > /proc/sys/net/ipv4/ip_forward`。

关键三：若 VPN Server 不是办公网络的默认网关，则想办法在默认网关上添加到 10.8.0.0/24 的路由项目，网关为服务器的内部 IP 地址。

关键一中的操作只能让客户端知道去往公司网络的包如何路由，而关键三的操作是为了让公司网络里的主机知道去往 VPN Client 的包如何路由。

其实关键三本来也很好实现，但由于我们的办公网络由美国总部的 network engineer 划分，我们向其申请时却被告知由于整个网络环境已规划好，防火墙不能随便添加静态路由，所用 OpenVPN 进行到这步时基本就告以失败了。

后来我尝试了一下在单网卡下配置 PPTPD VPN，很顺利就实现了点对局，特将此过程跟大家分享一下。

操作系统：Centos5.5 x86_64，IP：192.168.4.222，单网卡 eth0，服务器用的是 PowerEdge 2850。准备安装 PPTPD 服务，其过程如下：

1) 检查服务器是否有必要的支持。如果检查结果没有这些支持的话，是不能安装 PPTP 的。执行如下指令：

```
modprobe ppp-compress-18 && echo ok
```

这条命令执行后，显示 ok 则表示通过。不过接下来还需要做另一个检查，输入如下指令：

```
cat /dev/net/tun
```

如果这条指令显示结果为下面的文本，则表示通过：

```
cat: /dev/net/tun: File descriptor in bad state
```

2) 修改内核设置，使其支持转发。这步必须做，不然你仅仅能连接 VPN 机器，后面的局域网的资源均不能访问。编辑 `/etc/sysctl.conf` 文件如下：

```
vim /etc/sysctl.conf
```

将 `net.ipv4.ip_forward` 改为 1，变成下面的形式：

```
net.ipv4.ip_forward=1
```

保存退出，并执行下面的命令使之生效：

```
sysctl -p
```

3) 安装 PPTP。这个软件在 yum 源里是没有的，我们需要手动下载。先切换到 tmp 目录如下：

```
cd /tmp
```

然后执行下面的命令来下载 PPTPD 安装包，分别对应 32 位系统和 64 位系统，命令如下：

```
wget http://acelnmp.googlecode.com/files/pptpd-1.3.4-1.rhel5.1.i386.rpm
wget http://acelnmp.googlecode.com/files/pptpd-1.3.4-1.rhel5.1.x86_64.rpm
```

如果你的 Centos 是 32 位的，则执行 32 位的那条指令；如果是 64 位的 Centos，则执行 64 位的那条指令。注意不要搞错了。

接下来安装 PPTP，同样分 32 位和 64 位系统命令如下：

```
rpm -ivh pptpd-1.3.4-1.rhel5.1.i386.rpm
rpm -ivh pptpd-1.3.4-1.rhel5.1.x86_64.rpm
```

这里补充说明一点，由于我的办公网络就是单网卡环境，所以没必要开启 iptables 的 MASQUERADE 功能。因为我的 VPN 拨进来就可以利用防火墙的 NAT 功能上网，我特地用 <http://www.ip168.com> 观察了客户机出去的公网 IP，全被替换成防火墙的公网 IP 了。

4) 配置 PPTP。首先我们要编辑 /etc/pptpd.conf 文件，命令如下：

```
vim /etc/pptpd.conf
```

找到 localip 和 remoteip 这两个配置项，将前面的“；”注释符去掉，更改为你期望的 IP 段值。localip 表示服务器的 IP，remoteip 表示分配给客户端的 IP 地址，可以设置为区间。这里我们使用 PPTP 默认的配置如下：

```
localip 192.168.4.222
remoteip 192.168.4.242-246
```

注意 这里的 IP 段设置为 192.168.4.242 ~ 246，由于 PPTPD 分配给 VPN 拨号用户的 IP 都是真实的物理地址 IP，为了不与公司的原 IP 地址冲突，所以建议将这几个地址单独拿出来给 PPTPD VPN 使用。

5) 设置 PPTP VPN 账号密码。我们需要编辑 /etc/ppp/chap-secrets 这个文件如下：

```
vim /etc/ppp/chap-secrets
```

在这个文件里面，按照“用户名 pptpd 密码*”的形式编写，一行一个账号和密码。比如添加用户名为 test、密码为 1234 的用户，则编辑如下内容：

```
test pptpd 1234 *
```

其中第 4 列为 IP 地址，我们可以为特定用户手动指定特定 IP 或为“*”。

如果没有指定，为“*”，那么 PPTP VPN 服务器会从 /etc/pptpd.conf 文件中我们设定的 remoteip

中选择一个分配给客户端。

6) 重启 PPTP 服务。输入下面的指令重启 PPTP:

```
/etc/init.d/pptpd restart
```

我个人习惯用 `service pptpd restart`。

现在你已经可以连接自己的 VPN 并浏览网页了。不过我们还需要做最后的一步。

7) 设置开机自动运行服务。最后一步是将 PPTP 和 iptables 设置为开机自动运行, 这样就不需要每次重启服务器后手动启动服务了。当然你不需要自动启动服务的话可以忽略这一步。输入指令如下:

```
chkconfig pptpd on
```

记得让你的 Network Engineer 将 192.168.4.222 映射一个公网 IP, 如 220.249.xx.xx, 我做的是 DMZ 映射, 220.249.xx.xx 即是我们的客户机拨号时连接的公网 IP。

下面我们来思考一下客户机如何在 Windows XP 机上进行拨号连接呢?

前面配置了服务器端, 下面再简单说一下 Windows Server 2003 (Windows XP 类似) 客户端的 VPN 连接情况, 因为此操作都是图形化界面, 所以这里全部采用界面图来解说, 整个步骤如图 7-2 至图 7-7 所示。先选择你的网络邻居, 用右键点属性, 然后选择“创建一个新的连接”, 如图 7-2 所示 (其余步骤均由图示说明)。



图 7-2 客户端新建 VPN 拨号连接步骤一

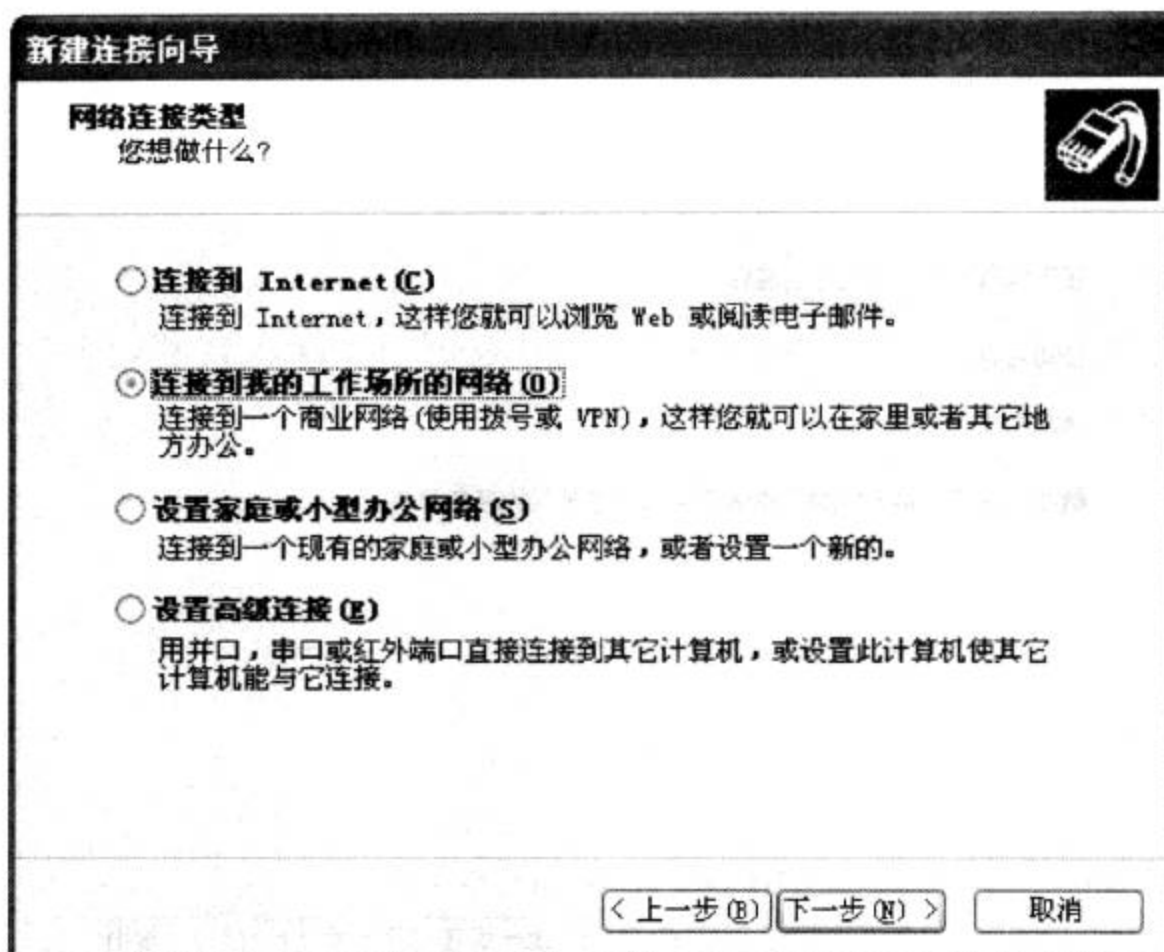


图 7-3 客户端新建 VPN 拨号连接步骤二

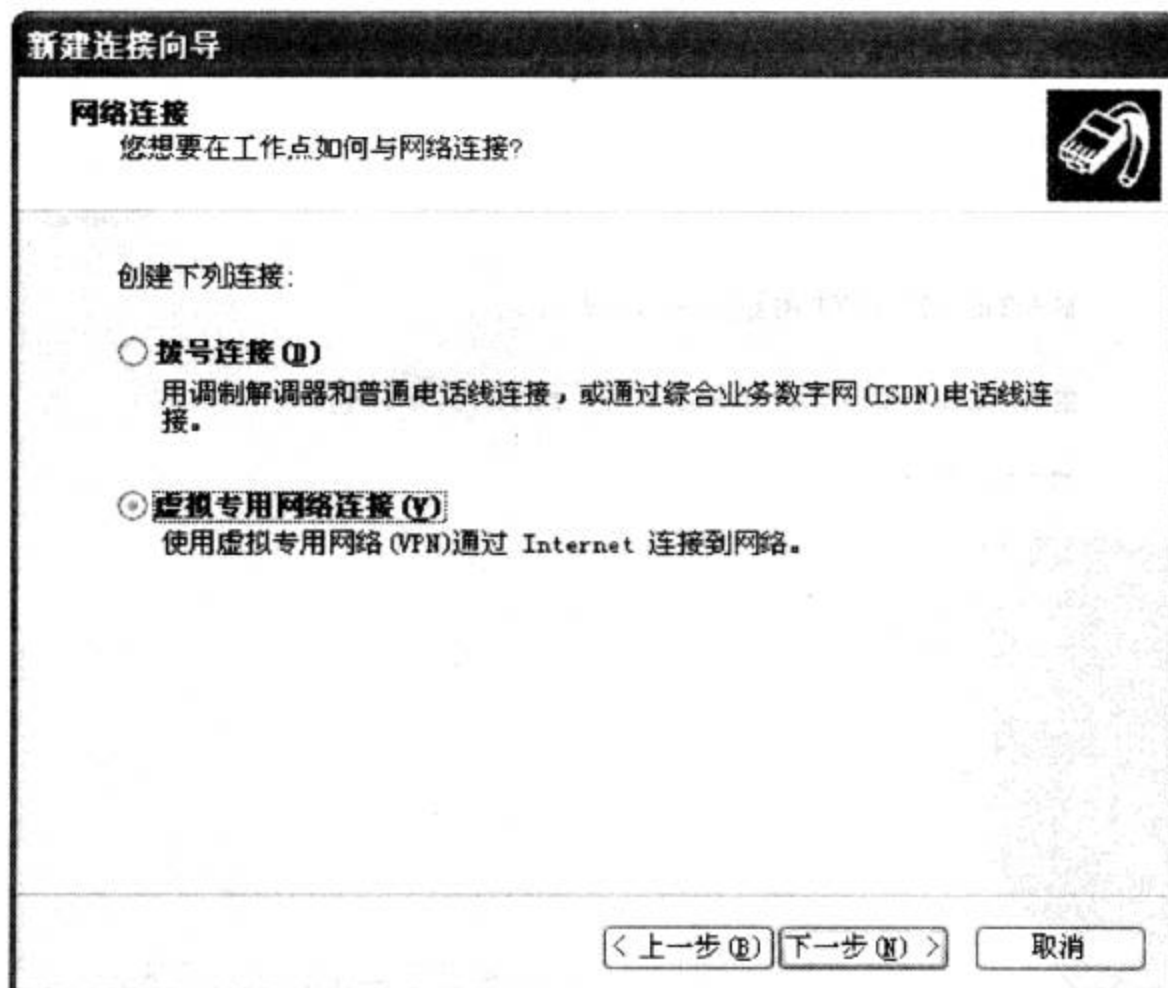


图 7-4 客户端新建 VPN 拨号连接步骤三

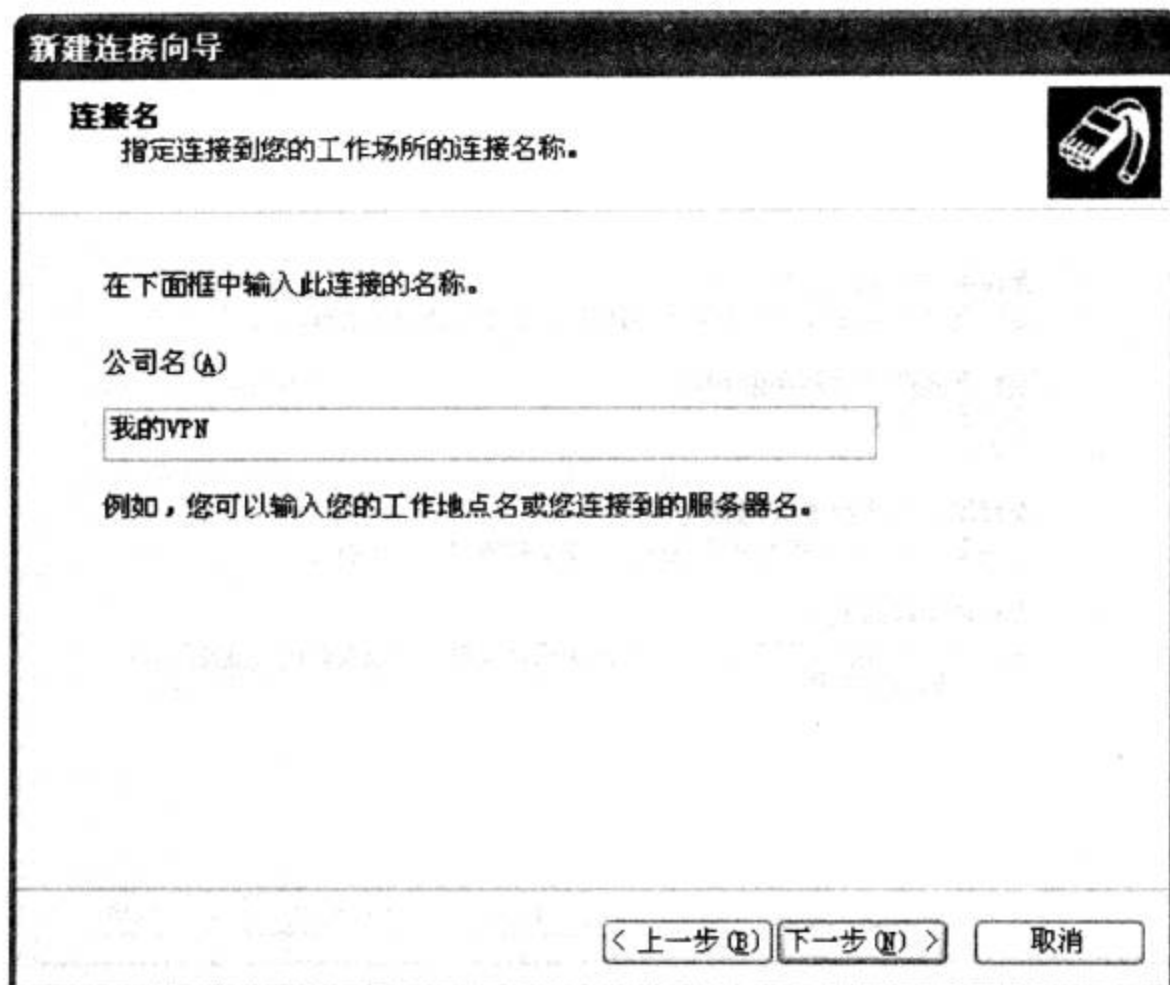


图 7-5 客户端新建 VPN 拨号连接步骤四

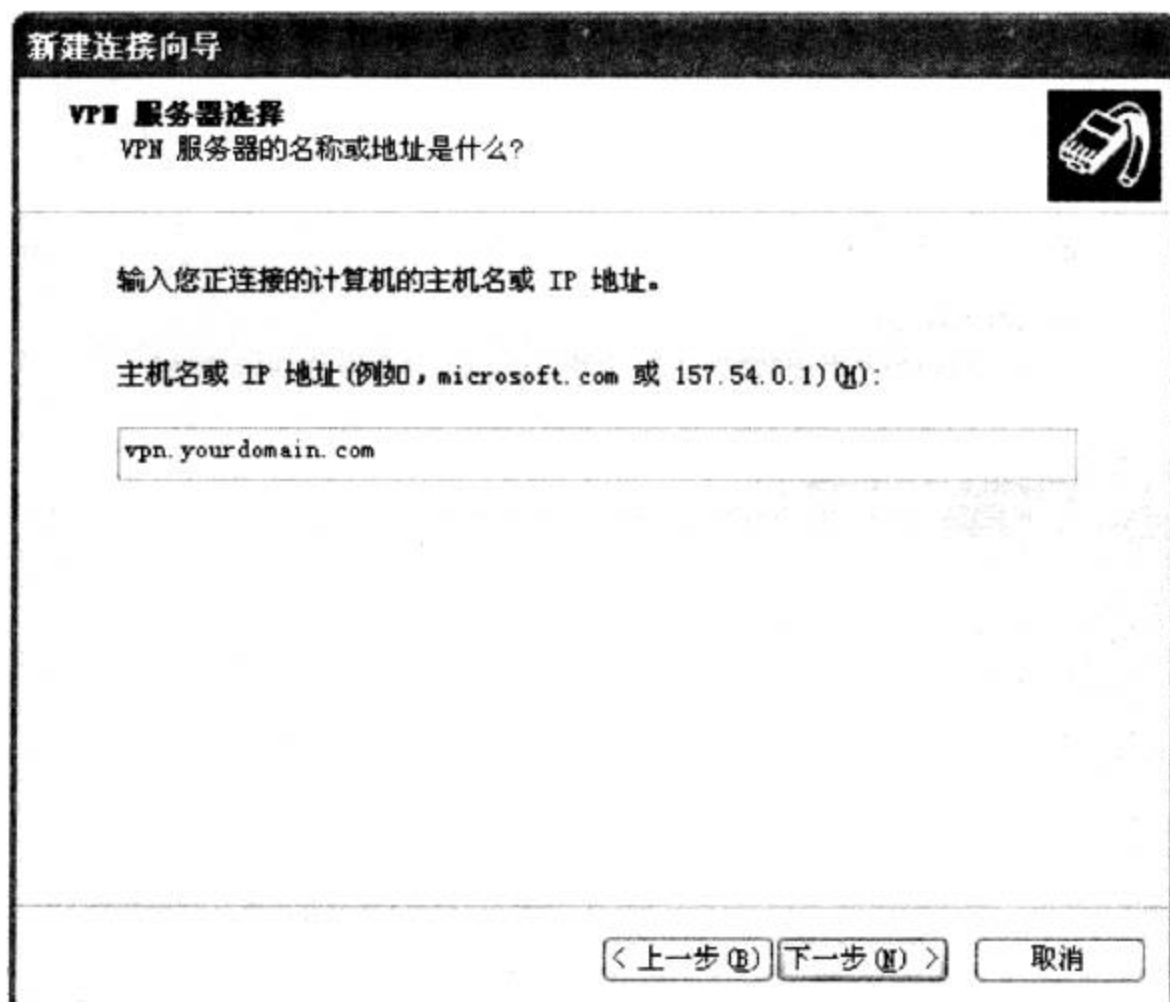


图 7-6 客户端新建 VPN 拨号连接步骤五

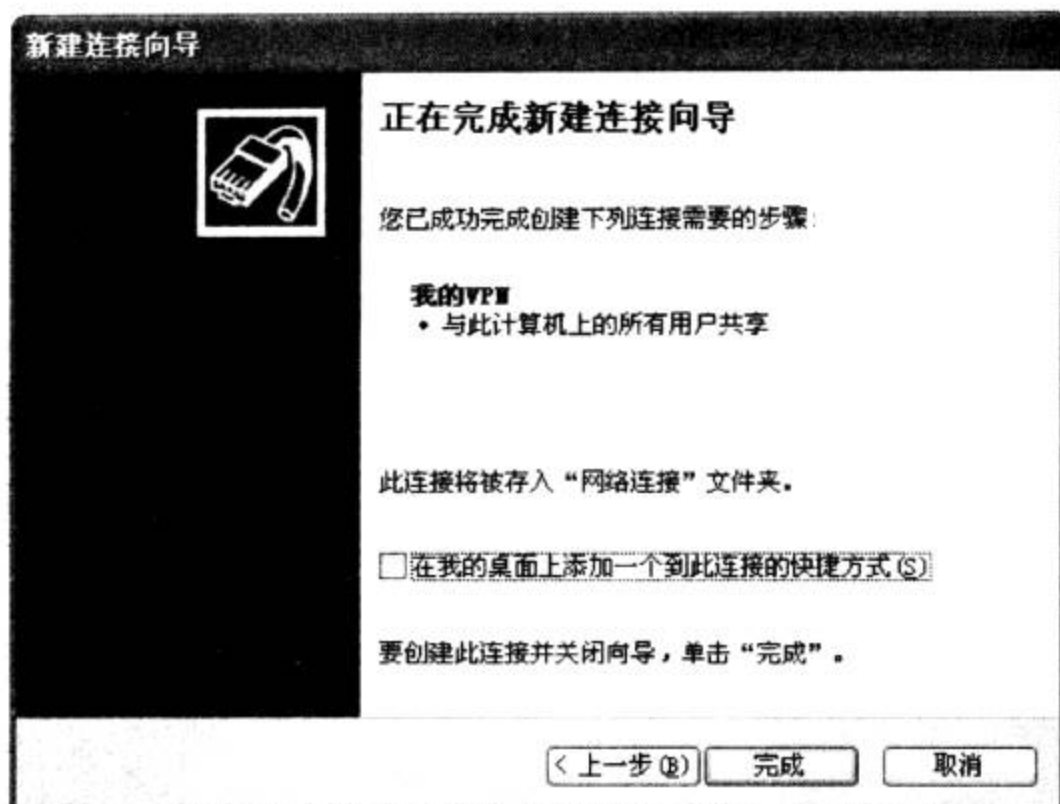


图 7-7 客户端新建 VPN 拨号连接步骤六

当我们从公网上用 VPN 拨号上来以后，我们就身处公司的办公环境，此时我们想远程访问桌面，或者连接公司内部 SVN 服务器、Samba 文件服务器或公司内部的 MySQL 数据库均可，此时的我们就相当于在公司的内网中了。此时尤其要注意安全问题，因为这个时候我们的 VPN 客户已经相当于在公司内部办公了，所以在病毒防护及权限控制方面都要注意。我其实比较推荐下面的做法：将防火墙多增加一块网卡，大家将要共享的资料服务器 FTP、Samba 及 Web 服务器放在 DMZ 区域，DMZ 的网段跟内部办公网络区分开来（即二者内部是不允许相互通信的），这样公司的办公网络就更为安全了，这种网络拓扑架构也简单，如图 7-8 所示。

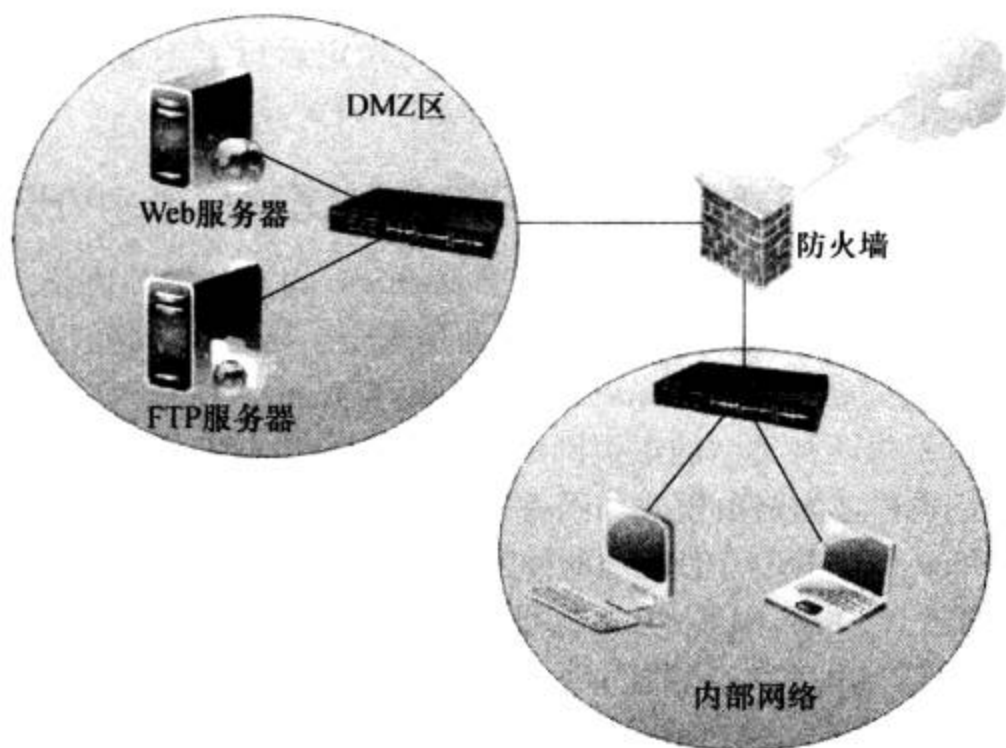


图 7-8 更安全的 DMZ PPTPD VPN 方案

下面跟大家分享一下 2002 年我在书店工作时用的 PPTPD VPN 方案：整个 VPN 架构很简单，将计算机数据处理中心作为数据统一中心，对图书的进货、销售、盘存、批发、打折数据库进行处理，然后跟物流和门市中心进行数据交互，门市之间与物流之间不进行交互。本着节约成本的考虑，计算机中心有固定的 IP，其余门市之间与物流中心之间只有 ADSL 拨号上网，如图 7-9 所示。

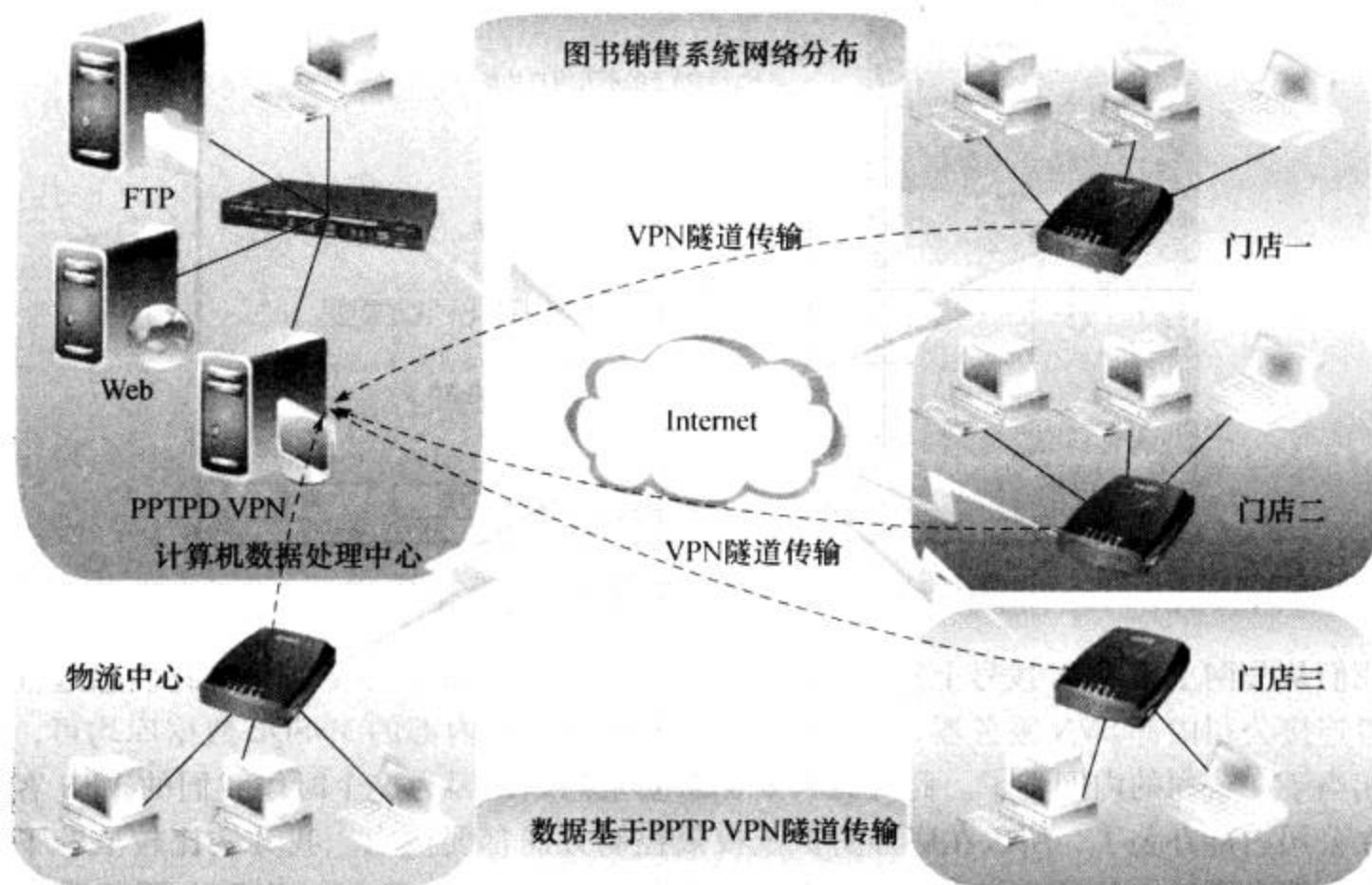


图 7-9 图书销售系统网络分布图

在 Linux 下构建 PPTPD VPN 服务器非常简单和方便，大家也可以在许多 VPN 系统上见到它的身影。目前我单独分配了一台联通 IP 的机器给 PPTPD VPN 作为公司网通用户的 VPN，它作为 OpenVPN 电信服务器的备份 PPTPD VPN 服务器，已在公司稳定运行半年多了，效果比较满意。由于它配置简单，又是免费开源的，所以我也向大家推荐这门 VPN 技术。

7.5.2 案例二：在 Centos5.5 x86_64 下路由模式配置 OpenVPN 服务器

OpenVPN 是一个开源的加密隧道构建工具，基于 OpenSSL 的 SSL/TLS 协议，可以在 Internet 中实现点对点的 SSL VPN 安全连接。使用 OpenVPN 的好处是安全、易用和稳定，且认证方式灵活，它具备实现 SSL VPN 解决方案的完整特性。OpenVPN 可以应用于 Linux、Unix、Mac OS 及 Windows 等各种操作系统平台。OpenVPN 提供两种类型的虚拟网络接口：TUN 和 TAP，分别用于建立 IP 隧道、以太网桥接，这两种模式我们也称之为路由模式和网桥模式。另外说明一下，网桥模式是不需要路由的，这点大家不要混淆。在 Linux/FreeBSD 中使用这两种虚拟设备，需要对应的内核模块支持。在 RHEL5.5、Centos5.5 及 FreeBSD8 系统（32 位和 64 位均可）中默认已编译好 TUN 模块，直接使用即可。

OpenVPN 的官方站点是 <http://OpenVPN.net>，目前稳定版为 OpenVPN-2.0.9，大家可以在官网上下载到很多跟 OpenVPN 相关的软件包。嫌麻烦的朋友可以直接在 51cto.com 网站下载，有热心的网友已经将 OpenVPN 相关的软件包打包，下载地址是：<http://down.51cto.com/data/194536#>。

我替一家公司设计 VPN 家庭办公方案时，初期部署的是 PPTPD VPN 方案，稳定性和加密性大

家都还满意。可后期在上线运作时发现，在用电信 ADSL 作 NAT 路由器的小区环境中，根本连接不了公司的公网 PPTPD VPN 服务器，拨号时出现了 619 报错，具体原因如下：

出现这种情况大多数是因为客户机连接 Internet 的网关（如家庭宽带路由，公司上网网关路由或防火墙）NAT-T 功能关闭或对 VPN 支持性不好，主要是对 GRE 及 PPTP 协议的 NAT-T 不支持。可若打开网关路由的 NAT-T 功能还是出现错误，则需要更换网关设备了，现在市面上大多数设备已经支持此功能，但以前的设备基本不支持。

鉴于这种情况，我考虑放弃 PPTPD VPN，选用 OpenVPN，它的穿透能力还是很强的。在这种局对局的环境中，OpenVPN 强大的穿透能力将会很好地发挥它的功效，这恰好也是 PPTPD VPN 不具备的功能。

值得注意的是，为了消除防火墙及路由器的影响，我前期直接将 OpenVPN 服务器放到了防火墙前面（如果置于防火墙后，还要考虑 DMZ 映射及路由方面的因素）。另外，考虑到稳定性需求，我用的是 Centos5.5 x86_64，LAN：192.168.4.222，WAN：220.249.x.x，后来经过测试发现，OpenVPN 放在防火墙后面效果会更好，稳定性和安全性也更有保证。所以后来还是采取了此种方案，将 OpenVPN 服务器置于防火墙后，只映射 1197 端口。

这里说明一下 OpenVPN 的网络部署应该注意的问题，如果你所在小区的局域网是 192.168.1.0，而你的 OpenVPN 所在的局域网也是 192.168.1.0 的话，那么就有问题了，你是拨不上 OpenVPN 服务器的。所以你在规划网络时，应该考虑不用 192.168.1.0 的网段，而是用 192.168.20.0 或 192.168.10.0 这些不常见的网段。你到星巴克咖啡馆边喝咖啡边上网时也许会发现，大多数提供类似无线服务的，基本上都是将局域网设计成了 192.168.10.0。所以如果公司也要上 VPN 环境的话，建议尽量不要用 192.168.1.0 的网段。

OpenVPN 的完整安装步骤如下（以下过程不仅可在 Centos5.5 下通过，在 RHEL5.5 及 FreeBSD8 或 8.1 下均测试通过，32 位及 64 位均可；我这里是测试环境，OpenVPN 用的是 192.168.1.0 网段的 IP，IP 地址为 192.168.1.101）。

1. 安装前的准备工作

工欲善其事，必先利其器。我们先做好 OpenVPN 安装前的准备工作，这样后面就会很顺利了。OpenVPN 是基于 OpenSSL 的，所以这里需要安装 OpenSSL。

在 FreeBSD8 下可采用 port 安装，命令如下：

```
cd /usr/ports/security/openssl && make install clean
```

在 RHEL5.5 | Centos5.5 下用 yum 安装，命令如下（最小化安装系统的朋友记得安装上 gcc 等软件包）：

```
yum -y install openssl openssl-devel gcc
```

软件包下载后放在 /usr/local/src 里，它们分别是：

lzo-2.03.tar.gz

OpenVPN-2.0.9-gui-1.0.3-install.exe

OpenVPN-2.0.9.tar.gz

在地址 <http://down.51cto.com/data/194536#> 下有它们的合集下载，想一个个下载的也行。

2. 安装 lzo 软件包

安装 lzo 软件包，用于压缩隧道通信数据以加快传输速度，操作步骤如下（这里也提供了 lzo

软件包的单独下载地址，见 wget 后面的地址）：

```
cd /usr/local/src
wget http://www.oberhumer.com/opensource/lzo/download/lzo-2.03.tar.gz
tar zxvf lzo-2.03.tar.gz
cd lzo-2.03
./configure --prefix=/usr && make && make install
```

3. 安装 OpenVPN-2.0.9

操作步骤如下：

```
tar zxvf OpenVPN-2.0.9.tar.gz
cd OpenVPN-2.0.9.tar.gz
./configure --with-lzo-lib=/usr && make && make install
```

这里是单独下载 OpenVPN 的地址：wget http://OpenVPN.net/release/OpenVPN-2.0.9.tar.gz。

4. OpenVPN 服务器端的配置

(1) 建立 CA 的详细信息

在 OpenVPN 源代码的目录下有一个 \easy-rsa\2.0 目录，进入后修改 vars 文件最后部分的信息，如下所示（大家根据各自的信息来改正）：

```
cd /usr/local/src/openvpn-2.0.9/easy-rsa/2.0
vim vars
export KEY_COUNTRY="CN"
export KEY_PROVINCE="BJ"
export KEY_CITY="Beijing"
export KEY_ORG="cn7788"
export KEY_OU="cn7788.com"
export KEY_EMAIL="yuhongchun027@163.com"
```

这里说明一下这些选项代表的含义如下：

```
export KEY_COUNTRY="CN"
```

定义你所在的国家，2 个字符为限。

```
export KEY_PROVINCE="BJ"
```

表示你所在的省份。

```
export KEY_CITY="Beijing"
```

表示你所在的城市。

```
export KEY_ORG="cn7788"
```

表示你所在的组织。

```
export KEY_OU="cn7788.com"
```

表示你的单位。

```
export KEY_EMAIL="yuhongchun027@163.com"
```

表示你的电子邮件地址。

接下来输入命令如下：

```
source vars
```

此命令输入后会显示如下结果：

```
NOTE:when you run ./clean-all, I will be doing a rm -rf on/etc/openvpn/easy-rsa/keys
```

接着输入下面的命令：

```
./clean-all
```

它用来初始化 keys 目录，创建所需要的文件和目录。如果生成了第一个客户端的证书文件，以后生成第二个客户端证书文件时这步就不需要执行了，它会清空所有的证书文件，请注意这一点。

然后输入以下命令：

```
./build-ca
```

它会生成 root CA 证书，用于签发 server 和 client 证书，密钥跟 OpenSSL 紧密结合，如果没有修改的地方一路回车即可，如下所示：

```
Generating a 1024 bit RSA private key
.....++++++
.....++++++
writing new private key to 'ca.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [CN]:
State or Province Name (full name) [BJ]:
Locality Name (eg, city) [Beijing]:
Organization Name (eg, company) [cn7788]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) [cn7788 CA]:cn7788 CA
Email Address [yuhongchun027@163.com]:

[root@localhost 2.0]# ls -lsart keys |grep ca
4 -rw----- 1 root root 887 May 21 16:32 ca.key
4 -rw-r--r-- 1 root root 1200 May 21 16:32 ca.crt
```

我们可以看到 ca.crt、ca.key 文件都已经生成了。

下面我们为服务器生成 Diffie-Hellman 文件，后面配置 OpenVPN Server 时需要用到此文件，命令如下：

```
[root@localhost 2.0]# ./build-dh
```

This is going to take a long time

★ ★ ★ ★ ★

```
./build-key-server server
```

入 y 然后回车, 其他可参照以下内容:

• • • • • + + + + +

.....+++++

— — — — —

into your certificate request.

What you are about to enter is what is called a Distinguished Name or a DN.

There are quite a few fields but you can leave some blank

For some fields there will be a default value.

If you enter '.', the field will be left blank.

— — — — —

Country Name (2 letter code) [CN]:

State or Province Name (full name) [BJ]:

Locality Name (eg, city) [Beijing]:

Organization Name (eg, company) [cn7788]:

Organizational Unit Name (eq. section) []:

```
Common Name (eg, your name or your server's hostname) [server]:server
```

Email Address [yuhongchun027@163.com]:

Please enter the following 'extra' attributes

to be sent with your certificate request

```
A challenge password [ ]:123456
```

An optional company name []: **cn7788.com**

Using configuration from /usr/local/src/OpenVPN-2.0.9/easy-rsa/2.0/openssl.cnf

Check that the request matches the signature

Signature ok

The Subject's Distinguished Name is as follows

countryName : PRINTABLE: 'CN'

```
stateOrProvinceName :PRINTABLE:'BJ'
```

```
localityName      :PRINTABLE:'Beijing'
```

```
organizationName      :PRINTABLE:'cn7788'
```

```
commonName      :PRINTABLE:'server'
```

emailAddress :IA5STRING:'yuhongchun027@163.com'

Certificate is to be certified until May 18 08:47:05 2021 GMT (3650 days)


```
Sign the certificate? [y/n]:y
```

```
1 out of 1 certificate requests certified, commit? [y/n]y
```

```
Write out database with 1 new entries
```

```
Data Base Updated
```

(3) 为客户端生成客户端证书文件

我这里以两个名字来举例说明，一个是 andrewy，另一个是 yuhongchun。在 OpenVPN 体系中，每一个登录的 VPN 客户端需要有一个证书，每个证书在同一时刻只能供一个客户端连接（如果有两个机器安装相同的证书，而且同时拨服务器，虽然都能拨上，但是只有第一个拨上的才能连通网络，第二个拨号时会每隔 5 秒钟就会断掉，这个时候就应该检查此证书是否已经被占用了），所以需要建立许多份证书。下面将建立两份证书，名称分别为 andrewy 和 yuhongchun，记得 hostname 名字不要一样，命令如下：

```
./build-key andrewy
```

```
Generating a 1024 bit RSA private key
```

```
.....++++++
```

```
.....++++++
```

```
writing new private key to 'andrewy.key'
```

```
-----
```

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
```

```
What you are about to enter is what is called a Distinguished Name or a DN.
```

```
There are quite a few fields but you can leave some blank
```

```
For some fields there will be a default value,
```

```
If you enter '.', the field will be left blank.
```

```
-----
```

```
Country Name (2 letter code) [CN]:
```

```
State or Province Name (full name) [BJ]:
```

```
Locality Name (eg, city) [Beijing]:
```

```
Organization Name (eg, company) [cn7788]:
```

```
Organizational Unit Name (eg, section) []:
```

```
Common Name (eg, your name or your server's hostname) [andrewy]:andrewy
```

```
Email Address [yuhongchun027@163.com]:
```

```
Please enter the following 'extra' attributes
```

```
to be sent with your certificate request
```

```
A challenge password []:123456
```

```
An optional company name []:cn7788.com
```

```
Using configuration from /usr/local/src/OpenVPN-2.0.9/easy-rsa/2.0/openssl.cnf
```

```
Check that the request matches the signature
```

```
Signature ok
```

```
The Subject's Distinguished Name is as follows
```

```
countryName :PRINTABLE:'CN'
```

```
stateOrProvinceName :PRINTABLE:'BJ'
```

```
localityName :PRINTABLE:'Beijing'
```

```
organizationName :PRINTABLE:'cn7788'
```

```
commonName :PRINTABLE:'andrewy'
```

```
emailAddress :IA5STRING:'yuhongchun027@163.com'
```

```
Certificate is to be certified until May 18 08:51:28 2021 GMT (3650 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

yuhongchun 用户的证书文件的生成跟以上类似，记得名字应不一样，命令如下：

```
./build-key yuhongchun
```

正常结束以上过程后，我们可以看到 keys 目录里生成了正常的证书文件，用以下命令可以查看：

```
ls -lsart keys
total 92
4 drwxrwxrwx 3 root root 4096 May 21 16:29 ..
4 -rw----- 1 root root 887 May 21 16:32 ca.key
4 -rw-r--r-- 1 root root 1200 May 21 16:32 ca.crt
4 -rw-r--r-- 1 root root 245 May 21 16:43 dh1024.pem
4 -rw----- 1 root root 887 May 21 16:47 server.key
4 -rw-r--r-- 1 root root 737 May 21 16:47 server.csr
4 -rw-r--r-- 1 root root 3848 May 21 16:47 server.crt
4 -rw-r--r-- 1 root root 3848 May 21 16:47 01.pem
4 -rw----- 1 root root 891 May 21 16:51 andrewy.key
4 -rw-r--r-- 1 root root 737 May 21 16:51 andrewy.csr
4 -rw-r--r-- 1 root root 3 May 21 16:51 serial.old
4 -rw-r--r-- 1 root root 209 May 21 16:51 index.txt.old
4 -rw-r--r-- 1 root root 20 May 21 16:51 index.txt.attr.old
4 -rw-r--r-- 1 root root 3728 May 21 16:51 andrewy.crt
4 -rw-r--r-- 1 root root 3728 May 21 16:51 02.pem
4 -rw----- 1 root root 887 May 21 16:54 yuhongchun.key
4 -rw-r--r-- 1 root root 741 May 21 16:54 yuhongchun.csr
4 -rw-r--r-- 1 root root 3735 May 21 16:54 yuhongchun.crt
4 -rw-r--r-- 1 root root 3 May 21 16:54 serial
4 -rw-r--r-- 1 root root 20 May 21 16:54 index.txt.attr
4 -rw-r--r-- 1 root root 317 May 21 16:54 index.txt
4 -rw-r--r-- 1 root root 3735 May 21 16:54 03.pem
4 drwx----- 2 root root 4096 May 21 16:54 .
```

(4) 修改 OpenVPN 服务器的配置文件/etc/server.conf

1) 复制示例文件到/etc 目录下，然后再在其基础下进行更改，命令如下：

```
cp -p /usr/local/src/openvpn-2.0.9/sample-config-files/server.conf /etc/server.conf
```

2) 编辑/etc/server.conf 文件，注意以下几个重点部分：

一是将 proto udp 改成 proto tcp，即服务启动用 TCP 1194 端口。

二是将 ca 那 4 行内容改成如下所示的形式（记得带上绝对路径地址）：

```
ca /usr/local/src/openvpn-2.0.9/easy-rsa/2.0/keys/ca.crt
```

ca 后面接的是 ROOT CA，它是使用 build-ca 生成的，用于验证客户端证书是否合法。

```
cert /usr/local/src/openvpn-2.0.9/easy-rsa/2.0/keys/server.crt
key /usr/local/src/openvpn-2.0.9/easy-rsa/2.0/keys/server.key
```

上面是 Server 端对应的证书等相关文件。

```
dh /usr/local/src/openvpn-2.0.9/easy-rsa/2.0/keys/dh1024.pem
```

以上是上面提到的所生成的 Diffie-Hellman 文件。

三是将 server 那行内容修改如下：

```
server 10.8.0.0 255.255.255.0
```

这就是 OpenVPN 服务器启动时为 VPN 网络分配的网段，注意不要与公司网络中的真实网段发生冲突。

四是将 verb3 改成 verb 5，我们可以多查看一些调试信息日志。

OpenVPN Server 配置文件的其他信息还有很多，有兴趣的朋友可自行参考 OpenVPN 的官方文档，为了将其过程简化，这里只突出重难点部分，没有一一罗列出来。

(5) 启动 OpenVPN 服务

- 1) 关闭 iptables 及 SELinux，以免对 OpenVPN 造成不必要的干扰。
- 2) 开启系统自身的 IP 转发功能，有了这个数据包才能在不同的网段之间流通。

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

3) 用命令行方式启动 OpenVPN 服务，由于 OpenVPN 服务启动后基本就不会作什么更改，所以我这里也没有编写脚本，也没有采用 OpenVPN 自身所带的脚本，命令如下：

```
/usr/local/sbin/openvpn --config /etc/server.conf
Sat May 21 17:47:31 2011 us=524051 Current Parameter Settings:
Sat May 21 17:47:31 2011 us=524364 config = '/etc/server.conf'
Sat May 21 17:47:31 2011 us=524470 mode = 1
Sat May 21 17:47:31 2011 us=524599 persist_config = DISABLED
Sat May 21 17:47:31 2011 us=524716 persist_mode = 1
Sat May 21 17:47:31 2011 us=524814 show_ciphers = DISABLED
Sat May 21 17:47:31 2011 us=524904 show_digests = DISABLED
Sat May 21 17:47:31 2011 us=525016 show_engines = DISABLED
Sat May 21 17:47:31 2011 us=525110 genkey = DISABLED
Sat May 21 17:47:31 2011 us=525217 key_pass_file = '[UNDEF]'
Sat May 21 17:47:31 2011 us=525313 show_tls_ciphers = DISABLED
...
Sat May 21 17:47:31 2011 us=584868 Listening for incoming TCP connection on [undef]:1194
Sat May 21 17:47:31 2011 us=585167 Socket Buffers: R=[87380 -> 131072] S=[16384 -> 131072]
Sat May 21 17:47:31 2011 us=585371 TCPv4_SERVER link local (bound): [undef]:1194
Sat May 21 17:47:31 2011 us=585500 TCPv4_SERVER link remote: [undef]
Sat May 21 17:47:31 2011 us=585619 MULTI: multi_init called, r=256 v=256
Sat May 21 17:47:31 2011 us=585775 IFCONFIG POOL:base=10.8.0.4 size=62
Sat May 21 17:47:31 2011 us=585909 IFCONFIG POOL LIST
Sat May 21 17:47:31 2011 us=586064 MULTI: TCP INIT maxclients=1024 maxevents=1028
Sat May 21 17:47:31 2011 us=586198 Initialization Sequence Completed
```

启动很顺利，若看到出现 Initialization Sequence Completed 的字样，表示启动成功了。我们可以

用 `lsof -i: 1194` 来验证一下, 命令如下:

```
lsof -i:1194
COMMAND PID USER  FD  TYPE DEVICE SIZE NODE NAME
OpenVPN 28208 root   5u  IPv4  31818      TCP *:OpenVPN (LISTEN)
```

(6) 安装客户端

在 Windows 客户端下安装 OpenVPN 的客户端程序并进行拨号。

1) 从 <http://OpenVPN.se/files/> 上下载与 OpenVPN 服务器版本一致的 Windows 客户端 OpenVPN GUI For Windows。

例如, 服务器装的是 OpenVPN 2.0.9, 那么下载的 OpenVPN GUI for Windows 应该是: OpenVPN-2.0.9-gui-1.0.3-install.exe。

2) 执行 OpenVPN-2.0.9-gui-1.0.3-install.exe, 一切采用默认的设置, 然后点 Next 直至安装结束。

3) 将 ca.crt、andrewy.crt、andrewy.key 复制到 C:\Program Files\OpenVPN\config(不同的用户使用不同的证书, 每个证书包括 .crt 和 .key 两个文件, 如 yuhongchun.crt 和 yuhongchun.key)。

4) 在 /usr/local/src/openvpn-2.0.9/sample-config-files/client.conf 的基础上建立客户端配置文件, 并将其改名为 C:\Program Files\OpenVPN\config\andrewy.ovpn, 这里为了避免麻烦, 已在 Windows XP 机器上安装了 gvim7.2, 我们可以用这款 Windows XP 下的 Vim 编辑软件来处理后缀名为 .ovpn 的文件。

□ 将 proto udp 改成 proto tcp。

□ 将 remote 那行内容改成如下内容:

```
192.168.1.101 1194
```

□ 将 ca 那 3 行内容改为如下内容:

```
ca ca.crt
cert client1.crt
key client1.key
```

□ 注释掉 comp-lzo。

5) 右击 Windows XP 右下角的 OpenVPN 图标, 选择 Connect。正常情况下应该能够连接成功, 并分配正常的 IP 地址 10.8.0.6, 如图 7-10 所示 (这里是 andrewy 的连接过程, yuhongchun 用户的连接过程以此类推)。

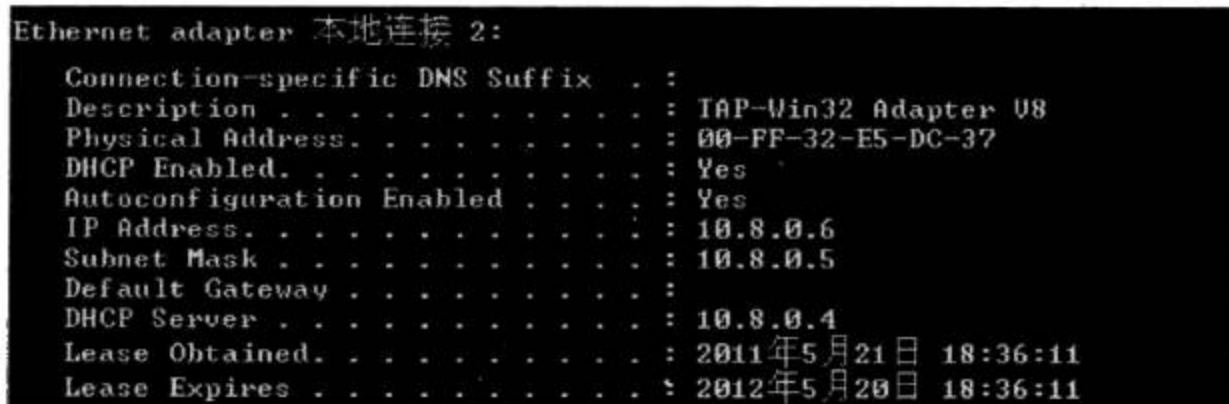


图 7-10 客户机正常分配到了 IP 地址

(7) 将内网 IP 映射为公网 IP

内网顺利连接之后, 我们记得要将这些 OpenVPN 的内网 IP 地址映射为公网 IP。为了安全起

见，这里只通过防火墙的 1194 端口映射出去即可。假设现在我们是在 192.168.4.0 的网段办公，OpenVPN 分配的 VPN 网段地址为 10.8.0.0、192.168.4.222，那我们该如何成功配置成局对局环境呢？我们可按照以下步骤配置：

1) 在 OpenVPN 的配置文件里增加 `push "route 192.168.4.0 255.255.255.0"`，目的是为客户端加一条路由，这样客户端才有可能访问到办公网络中除 VPN Server 之外的其他主机。有很多 VPN 客户端是直接添加默认路由的，这样的客户端所有连接请求都会被路由到 VPN 通道内，这会导致客户端此时不能访问 VPN，而此项添加指定地址的路由不会出现这一问题。

2) 在 OpenVPN 上开启 IP 转发，命令如下：

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

3) 在公司的网关或路由器上增加一条路由，添加到 10.8.0.0/24 的路由项目中，其网关为服务器的内部 IP 地址，目的是为了公司局域网里的主机知道去往 VPN Client 的包如何路由。记住，路由是双向和回环的，既要有来的路由，也应该有回的路由。如果公司的网络环境不允许更改，则可以在要被访问的机器上增加一条路由，命令如下：

```
route add 10.8.0.0 mask 255.255.255.0 192.168.4.222
```

就这样，OpenVPN 的局对局环境就配置成功了，以后无论是在小区环境中，还是在广电网及电信 ADSL 共享上网的网络环境中，都能很轻松地访问 OpenVPN 所在局域网内的资源。

下面介绍一下通过路由模式配置 OpenVPN 应该注意的问题。

每个人的网络拓扑不一样，如何使 OpenVPN 的客户端能够很轻松地访问除 OpenVPN 外的公司局域网内的机器呢？相信这个问题是用 OpenVPN 的朋友最为关注的，这个问题我们如何解决呢？大家可以参考以下几个办法：

- 可以把 OpenVPN 服务器也设成路由器，并使 OpenVPN 服务器作为这些需要被访问的内网机器的路由器，这样路由器和 OpenVPN 就在同一台机器上了。
- 如果内网中有多台机器需要被访问，那么可以在路由器上设置静态路由表，设置 10.8.0.0 网段的信息路由到 OpenVPN 服务器。具体设置参看自己路由器上的设置。
- 如果改不了路由器，或者内网需要修改的机器不多的话，也可以直接在内网要被访问的机器上执行 `route add 10.10.0.0 mask 255.255.255.0 192.168.4.222`（OpenVPN 服务器的 IP）命令（我的 Windows 服务器操作系统是 Windows Server 2003），来直接给内网的机器添加路由。这样当碰到 10.10.0.0 网段的信息时，它就会直接路由到 OpenVPN 服务器，而不会走路由器那条路了。

注意 一台机器只能用一个证书拨号，如果是两台机器都要用一个证书来拨号的话，第二台机器会发生每 5 秒就掉线的情况。虽然可以在 OpenVPN 服务器上通过配置来更改，但我建议还是一台机器使用一个证书比较好。

另外一个小问题就是：在进行证书制作工作时，仍旧需要进行初始化，但只需要进入 `OpenVPN\easy-rsa` 目录，运行 `vars` 就可以了，不需要 `./clean-all` 步骤，它会清除一切证书文件的，这一定要注意。

以上就是我通过路由配置 OpenVPN 服务器时的一些经验总结和心得，希望能给大家带来一些

帮助。每个人所在公司的网络情况不一样，别人的未必适合你，所以建议多动手，多尝试，毕竟实践出真知。相信你在部署整个 VPN 办公环境的时候，会发现 OpenVPN 的稳定和穿透能力相当强悍，我们在考虑 VPN 部署方案应该多利用它的这项特点。

7.5.3 案例三：在 FreeBSD8 x86_64 下网桥模式配置 OpenVPN 服务器

公司的网络拓扑说明如下：

整个公司划分为 3 大区域，总部在美国，武汉和成都都属于分部，代码 SVN、WIKI、项目管理服务器在武汉和成都均有。公司的重要业务服务器都放在美国机房里，网段是 192.168.10.0/24，10.1.0.0/24；武汉和成都的网段也比较多，武汉这边有 192.168.4.0/24，属于我们的办公网络，SVN、WIKI、项目管理等服务器放在 192.168.21.0/24 网段。成都的规模较小，只有 172.16.0.0/24 网段。平时 3 个办公区域都是通过防火墙 VPN 相连接的，如图 7-11 所示。

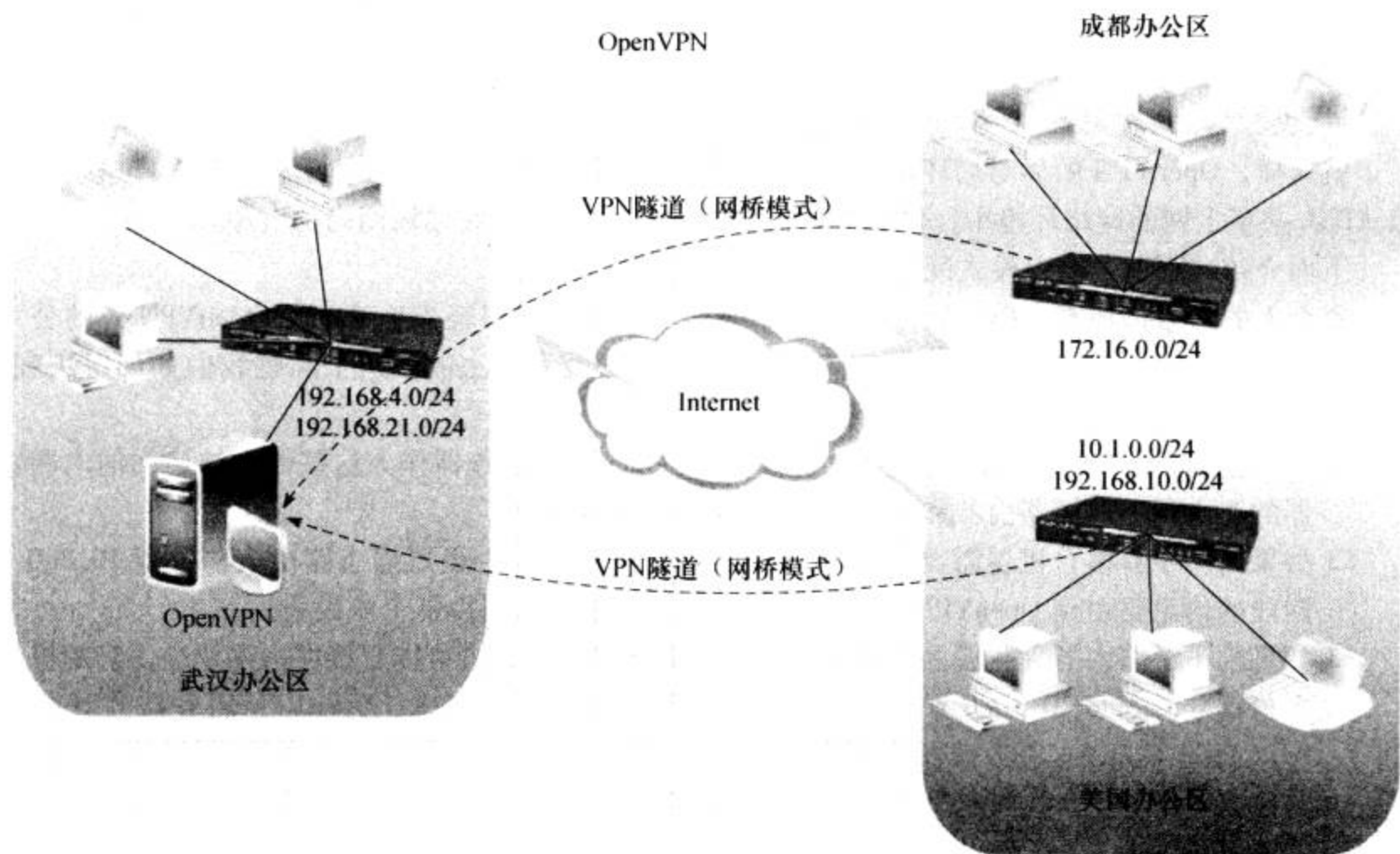


图 7-11 跨区域办公环境

同事们的办公需求如下：

公司的办公网络是 192.168.4.0/24，均只用单网卡 eth0，通过 Juniper 防火墙 SNAT 上网，即内网内所有机器的网关均是固定的 Juniper 防火墙，如 192.168.4.3 和 192.168.4.1，因为公司有两条专线，一条是电信的，一条是联通的，所以对应的防火墙也有两台。目前想通过外网拨号（VPN 服务器 IP 为 192.168.4.46）到公司内部局域网进行办公，另外还要求能够连通到公司内部 192.168.10.0、192.168.20.0、192.168.21.0，还有 10.1.0.0 网段的服务器（因为有的网段分配给

了 SVN 开发服务器，有的分配给了网站系统，在系统代码有问题时需要人工调试)。此要求比之前复杂多了。而且同事们的拨号环境不一样，有的是小区长城宽带，有的是 ADSL NAT 路由，还有的是电信 3G 无线上网。我将大家的需求归纳了一下，即：

- 要求能够在 ADSL NAT 路由或小区环境下顺利通过 OpenVPN 连接公司不同网段的服务器，即要求是局对局环境。
- 公司的办公环境是 192.168.4.0，拨上来要求能够连到 192.168.21.0、192.168.20.0、192.168.10.0 的服务器网段，这比以往的环境要求复杂。
- 公司的网络环境已定型，不可能在路由器或防火墙上做更改，所以想采用 OpenVPN 的路由模式是不可能的。
- 还要考虑到路由穿透的问题，所以放弃 PPTPD VPN 模式。

综合以上因素，最终我们决定采取 OpenVPN 的网桥模式来架设 OpenVPN 服务器。

在架设 OpenVPN 服务器之前，有个问题请大家思考一下：OpenVPN 的网桥和路由模式到底有什么不同？为了弄清楚这个问题，我们来一步步地分析（以下部分内容参考了 chinaunix 网友温占考的文章，特此注明）。

1) 我们该如何决定是选用路由还是桥接的 VPN？

总的来说，路由对大多数人来说是一种更好的方式，因为它比桥接效率更高也更容易设置。路由还可以给每个客户端设置不同的访问权限。

我也推荐使用路由，除非你需要使用依赖于桥接的特性，比如：

- VPN 需要处理非 IP 协议，例如 IPX 协议。
- 在 VPN 上运行的程序依赖于网络广播（例如局域网游戏）。
- 不建立 Samba 或 WINS 服务器，而允许在 VPN 上浏览 Windows 共享文件。

2) TUN 设备和 TAP 设备有什么不同呢？

TAP 设备是一块虚拟的以太网卡，TUN 设备是一个虚拟的点到点 IP 连接。

3) 什么是桥接呢？

桥接是在一个子网上创建一个虚拟的、广域的以太网 LAN 的一种技术。

桥接的实践信息请参考 Ethernet Bridging Mini-HOWTO。如果形象地解释那它就是连通不同局域网的桥梁，当外网用户 VPN 过来时，如果能够连通 OpenVPN 机器，那么外网用户都应该能够连通 OpenVPN 机器所连通的机器。

4) 桥接和路由有什么不同？

桥接和路由是通过 VPN 连接系统的两种方法。

桥接的优点是广播可以穿越 VPN——允许依赖局域网广播的软件运行，比如 Windows 的 NetBIOS 文件共享和网上邻居。

另外桥接无需配置路由，可以和以太网上的任何协议一起工作，包括 IPv4、IPv6、Netware IPX、AppleTalk 等。

而桥接的缺点是比路由效率低，扩展性不太好。

路由的优点是高效和可扩展，具有更好的 MTU 调节。

路由的缺点如下：

- 要使跨越 VPN 的网络浏览工作，客户端必须使用一个 WINS 服务器（比如 Samba）。

- 必须设置连接每一个子网的路由。
- 依赖于广播的软件不能看到在 VPN 另一边的机器。
- 仅支持 IPv4，只有连接两边的 tun 驱动明确支持 IPv6，才能支持 IPv6。

5) 桥接和路由在配置上有什么不同?

当客户端通过桥接方式连接远端网络时，它被分配一个远端物理以太网的 IP 地址，从而能够与远端子网的其他机器交互，就像它是连接在本地一样。桥接需要用特殊的 OS-相关的工具来将物理以太网卡和虚拟的 TAP 设备桥接起来。在 Linux 下，brctl 就是这样一个工具。

对于 Windows XP 或更高版本，可从“控制面板→网络连接”中选中 TAP-Win32 网卡和以太网卡，点击鼠标右键，选择桥接。而客户端通过路由方式连接时，它使用自己的独立子网，并且在客户机和远端网关上都设置了路由，从而使数据包无缝地穿越 VPN。客户端可以不只是一台机器，它可以是几台机器组成的一个子网。

桥接和路由很相似，主要的不同是路由的 VPN 不传送 IP 广播包，但是桥接的 VPN 传送 IP 广播包。

要使用桥接方式，连接的两端都必须使用 `--dev tap`，如果使用路由方式，可以使用 `--dev tap` 也可以使用 `--dev tun`，但是连接的两端必须一致。对于路由方式而言，`--dev tun` 的效率要更高一些。

下面来看一下桥接概览：

以太网桥接将一个以太网接口和一个或多个虚拟 TAP 接口结合 (combine)，并将它们桥接为一个桥接接口。以太网桥接代表一个物理以太网交换机 (switch) 的软件模拟，以太网桥可以认为是在一台机器上共享一个 IP 子网连接多个以太网卡 (物理的或虚拟的) 的软件交换机，通过将不同地方的一个物理以太网卡和一个 OpenVPN 的 TAP 接口桥接，可以将两个以太网在逻辑上合并为好像一个以太网一样。

在 FreeBSD8 x86_64 下以桥接模式配置 OpenVPN 服务器的详细步骤如下：

1) 说明一下 OpenVPN 的操作系统及网络分配等情况。

OpenVPN 服务器用的是 DELL 2950 服务器，采用 FreeBSD8 x86_64，分配的 IP 为 192.168.4.46，只连接一个网卡 eth0 (服务器本身自带了双网卡)，通过防火墙 SNAT 上网，网关为 192.168.4.3。在 FreeBSD8 下配置 br0 网卡相对 Linux 而言更为简单 (这也是我们选择 FreeBSD8 作为 OpenVPN 服务器的原因之一)，用脚本来添加，脚本 `/root/addbr0.sh` 的内容如下：

```
#!/bin/sh
kldload if_tap
ifconfig bridge0 create
ifconfig bridge0 addm re0
ifconfig bridge0 192.168.4.241 netmask 255.255.255.0 broadcast 192.168.4.255
/usr/local/sbin/OpenVPN --config /usr/local/etc/server.conf --daemon OpenVPN
ifconfig bridge0 addm tap0

ifconfig bridge0 deletem tap0
/usr/local/sbin/OpenVPN --config /usr/local/etc/server.conf --daemon OpenVPN
ifconfig bridge0 addm tap0
```

此脚本在编写测试时发现，`ifconfig bridge0 addm tap0` 必须填写两次才能正确添加，不知道这算不算 FreeBSD8 的一个 Bug。大家在查看此脚本代码时也不要诧异，这里并不是语法错误或写重复

了, 以上脚本我是直接从服务器上复制下来的。

另外要说明的是, 给 bridge0 分配的也是 192.168.4.0 网段的 IP, 我看到网上大部分的文章都是采用的相同 IP, 即 192.168.4.46, 建议另外用一个新的 IP 配置, 即 192.168.4.241, 这个 IP 不要跟局域网的其余 IP 发生冲突。脚本成功运行后, 可用命令 ifconfig 检查一下, 成功的配置应该如图 7-12 所示 (等内网测试顺利后, 直接映射此 bridge0-192.168.4.241 的 IP 地址为公网 IP)。

```

en0: flags=8802<BROADCAST,SIMPLEX,MULTICAST> metric 0 mtu 1500
    options=9b<RXCSUM, TXCSUM, VLAN_MTU, VLAN_HWTAGGING, VLAN_HWCSUM>
    ether 00:1b:21:59:df:da
    media: Ethernet autoselect
    status: no carrier
plip0: flags=8810<POINTOPOINT,SIMPLEX,MULTICAST> metric 0 mtu 1500
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> metric 0 mtu 16384
    options=3<RXCSUM, TXCSUM>
    inet6 fe80::1%lo0 prefixlen 64 scopeid 0x6
    inet6 ::1 prefixlen 128
    inet 127.0.0.1 netmask 0xff000000
bridge0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
    ether c2:fd:b1:f7:13:cf
    inet 192.168.4.241 netmask 0xfffff00 broadcast 192.168.4.255
    id 00:00:00:00:00:00 priority 32768 hellotime 2 fwddelay 15
    maxage 20 holdcnt 6 proto rstp maxaddr 100 timeout 1200
    root id 00:00:00:00:00:00 priority 32768 ifcost 0 port 0
    member: tap0 flags=143<LEARNING,DISCOVER,AUTOEDGE,AUTOPTP>
        ifmaxaddr 0 port 8 priority 128 path cost 2000000
    member: re0 flags=143<LEARNING,DISCOVER,AUTOEDGE,AUTOPTP>
        ifmaxaddr 0 port 3 priority 128 path cost 200000
tap0: flags=8943<UP,BROADCAST,RUNNING,PROMISC,SIMPLEX,MULTICAST> metric 0 mtu 1500
    ether 00:bd:23:fa:04:00
    inet 192.168.4.241 netmask 0xfffff00 broadcast 192.168.4.255
    Opened by PID 1316
[root@openvpn ~]#

```

图 7-12 正常添加 bridge0 后的显示

2) 安装过程简略带过, 安装的方法及配置证书过程可参考上一个案例。Server 端的配置文件详细如下, 这是重点部分, 相关语法我会详细介绍。下面的命令用来忽略掉配置文件中的注释部分及空行, 只取有效部分。

```
/usr/local/etc/server.conf | grep -v "^#" | grep -v "^;" | sed '/^$/d'
```

显示内容如下所示:

```

port 1194
proto udp
dev tap0
ca /usr/local/src/OpenVPN-2.0.9/easy-rsa/2.0/keys/ca.crt
cert /usr/local/src/OpenVPN-2.0.9/easy-rsa/2.0/keys/server.crt
key /usr/local/src/OpenVPN-2.0.9/easy-rsa/2.0/keys/server.key
dh /usr/local/src/OpenVPN-2.0.9/easy-rsa/2.0/keys/dh1024.pem
ifconfig 192.168.4.241 255.255.255.0
server -bridge 192.168.4.241 255.255.255.0 192.168.4.244 192.168.4.246
push "redirect -gateway"

```

```

client-to-client
keepalive 10 120
comp-lzo
user nobody
group nobody
persist-key
persist-tun
status OpenVPN-status.log
log-append OpenVPN.log
verb 5

```

以上脚本运行时已成功启动了 OpenVPN 服务,并分配了网桥 bridge0 地址为 192.168.4.241,可用命令 `lsof -i:1194` 来检验,结果如图 7-13 所示。

```

ifconfig bridge0 delnetm tap0
/usr/local/sbin/openvpn --config /usr/local/etc/server.conf --daemon openvpn
ifconfig bridge0 addm tap0

You have new mail in /var/mail/root
[root@openvpn ~]# lsof -i:1194
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
openvpn 1320 nobody 4u IPv4 0xfffff0001bde3a0 0t0 UDP *:1194

```

图 7-13 用 `lsof` 命令来验证 OpenVPN 是否正常启动

这里附上 OpenVPN 服务器的配置文件 `/usr/local/etc/server.conf` 的详细参数说明,大家可以对照配置文件来查看,如下所示:

```

;locala.b.c.d      #申明本机使用的 IP 地址,也可以不说明
port1194           #申明使用的端口,默认 1194
;prototcp          #申明使用的协议,默认使用 UDP,如果使用 HTTP proxy,必须使用 TCP 协议
proto udp
dev tap            #申明使用的设备,可选 tap 和 tun, tap 是二层设备,支持链路层协议, tun 是 IP 层的点
                  #对点协议,限制稍微多一些。本人习惯使用 TAP 设备,这个位置写成 tap 或 tap0 均可,大
                  #家也可以多尝试一下
ca /user/local/src/OpenVPN-2.0.9/easy-rsa/keys/ca.crt
#OpenVPN 使用的 ROOT CA 是使用 build-ca 生成的,用于验证客户证书是否合法
cert /usr/local/src/OpenVPN-2.0.9/easy-rsa/keys/server.crt
#Server 使用的证书文件
key /usr/local/src/OpenVPN-2.0.9/easy-rsa/keys/server.key
#Server 使用的证书对应的 key,注意文件的权限,防止被盗
dh /user/local/src/OpenVPN-2.0.9/easy-rsa/keys/dh1024.pem
#上面提到的生成的 Diffie-Hellman 文件
ifconfig-pool-persistipp.txt
#防止 OpenVPN 重新启动后“忘记”Client 端曾经使用过的 IP 地址
server-bridge 192.168.4.241 255.255.255.0 192.168.4.244 192.168.4.246
#此乃 Bridge 状态下类似 DHCPD 的配置,为客户分配地址,由于这里工作在桥接模式,所以我没有贴上 OpenVPN 在路
#由模式下的相关配置信息。OpenVPN 中有个非常有用的参数是 push,它表示通过 VPN Server 往 Client push 路由,
#Client 通过 pull 指令获得 Server push 的所有选项并应用之,但在网桥模式下就不需要对此进行配置了,因为
#push 指令用于路由模式
client-to-client
#如果可以让 VPN Client 之间相互访问,直接通过 OpenVPN 程序转发

```

```

keepalive 10 120
#NAT 后面使用 VPN,如果 VPN 长时间不通信,NAT Session 可能会失效,导致 VPN 连接丢失。为了防止此类事情发生,
  Keepalive 提供一个类似于 ping 的机制,此命令表示每 10 秒通过 VPN 的 control 通道 ping 对方,如果连续 120 秒
  无法 ping 通,则认为连接丢失,并重新启动 VPN,重新连接(对于 mode server 模式下的 OpenVPN 不会重新连接)
comp -lzo
#对数据进行压缩,注意 Server 和 Client 一致
max -clients 100
#定义最大连接数
user nobody
group nobody
#定义运行 OpenVPN 的用户组和用户
persist -key
#通过 Keepalive 检测超时后,重新启动 VPN,不重新读取 keys,保留第一次使用的 keys
persist -tun
#通过 Keepalive 检测超时后,重新启动 VPN,一直保持 tun 或 tap 设备是 linkup 的,否则网络连接会先 linkdown 然
  后 linkup
status OpenVPN -status.log
#定期把 OpenVPN 的一些状态信息写到文件中,以便自己写程序计时或进行其他操作
log
OpenVPN.log
#记录日志,每次重新启动 OpenVPN 后删除原有的 log 信息
verb 5
#相当于 debug level,即日志记录的级别,如果是 5 的话可以查看到更多的日志信息

```

客户端的配置文件较简单,这里就不详细阐述了,如下所示:

```
cat client.txt | grep -v "^;" | grep -v "^#" | sed '/^$/d'
```

有效的配置内容显示如下:

```

client
dev tap0
proto udp
remote IP
resolv -retry infinite
nobind
persist -key
persist -tun
ca ca.crt
cert andrewy.crt
key andrewy.key
comp -lzo
verb 5

```

IP 地址为 bridge0 通过防火墙映射出去的公网 IP。

注意 如果大家的客户端是 Windows 7 的话,建议就不要采用 OpenVPN-2.0.9-gui-1.0.3-install.exe 作为其 OpenVPN 客户端了,可以采取更高级版本的 OpenVPN-gui,不然是拔不上号的。WindowsXP_SP3 相对而言是客户系统中最为稳定的。

测试结果:

我们测试时分配的是联通的公网 IP，在测试过程中发现只要一拨号，无论是 ADSL 还是小区环境，只要能连通 192.168.4.46（另一个 IP 地址为 192.168.4.241），所有应连通的网段均能连通。虽然我只需要 SSH 环境即可完成日常工作，但其他同事需要远程桌面环境，考虑到大多数的上网宽带环境都是基于电信和长城宽带的，所以将其更改为电信的公网 IP 了。整个 OpenVPN 网桥办公环境已经稳定运行半年，而且非常快速，没有发生过丢包现象。在测试中我们也发现，如果客户端是二次 NAT 上网环境，则不能使用 OpenVPN，不过这样的上网环境并不多见。

注意 OpenVPN 还有许多细节问题，比如有的同事只想用 VPN 连进公司的网络时采用公司的内部 DNS，而在家办公时想一边工作一边聊 QQ、看新闻，这个时候我们就需要在 OpenVPN 的服务器端配置 DNS 不推送了，即不选择 push dns 这一项。另外，如何通过 OpenVPN 客户端的配置增加路由也是需要考虑的问题，虽然这些细节问题我在工作中都处理了，但由于每个人的网络环境不一样，我担心写出来会误导大家，所以建议大家通过自己的真实环境来领会和掌握。

7.6 部署 OpenVPN 服务器的注意事项

7.6.1 OpenVPN 如何注销用户

公司有同事离职，需注销其 VPN 证书，但总不能因为一个同事的离职而将所有的证书文件重新生成一遍吧。其实 OpenVPN 在设计时就考虑到了这一点，那就是使用 revoke-full 命令注销相关证书。执行命令进入 OpenVPN 的安装目录，在 easy-rsa 目录下，即 cd/usr/local/src/openvpn-2.0.9/easy-rsa 下使用 revoke-full 命令来注销其证书，命令如下：

```
./revoke - full andrewy
Using configuration from/usr/local/src/OpenVPN - 2.0.9/easy - rsa/2.0/openssl.cnf
error on line 282 of config file'/usr/local/src/OpenVPN - 2.0.9/easy - rsa/2.0/openssl.cnf'
1536:error:0E065068:configuration file routines:STR_COPY:variable has no
value:conf_def.c:629:line 282
Using configuration from/usr/local/src/OpenVPN - 2.0.9/easy - rsa/2.0/openssl.cnf
error on line 282 of config file'/usr/local/src/OpenVPN - 2.0.9/easy - rsa/2.0/openssl.cnf'
1537:error:0E065068:configuration file routines:STR_COPY:variable has no
value:conf_def.c:629:line 282
cat:crl.pem: No such file or directory
andrewy.crt: /C = CN/ST = BJ/L = Beijing/O = cn7788/CN = andrewy/emailAddress = yuhongchun027@
163.com
error 3 at 0 depth lookup:unable to get certificate CRL
```

我们会很惊奇地发现，OpenVPN 报错了，别急，这是 OpenVPN 自身的 Bug。解决这个 Bug 的方法其实也很简单，即注释掉/usr/local/src/openvpn-2.0.9/easy-rsa/2.0/openssl.cnf 文件最后的几行，如下所示：

```
#[ pkcs11_section]
#engine_id = pkcs11
#dynamic_path = /usr/lib/engines/engine_pkcs11.so
```



```
#MODULE_PATH = $ENV::PKCS11_MODULE_PATH
#PIN = $ENV::PKCS11_PIN
#init = 0
```

然后再执行如下命令：

```
revoke -full andrewy
```

值得注意的是，请保持 OpenVPN 的运行状态，不要在关闭状态下执行以上操作，那样是不会成功的。命令执行后会有如下显示：

```
Using configuration from /usr/local/src/OpenVPN-2.0.9/easy-rsa/2.0/openssl.cnf
Revoking Certificate 02.
Data Base Updated
Using configuration from /usr/local/src/OpenVPN-2.0.9/easy-rsa/2.0/openssl.cnf
andrewy.crt:/C = CN/ST = BJ/L = Beijing/O = cn7788/CN = andrewy/emailAddress = yuhongchun027@
163.com
error 23 at 0 depth lookup:certificate revoked
```

这个时候，如果 OpenVPN 里有 error23 字样，就是表示已经注销成功了。不过你会发现这个时候用这个证书还可以登录。这是因为上面的操作在 keys 下产生了 `crl.pem`，里面就是注销掉的证书，也就是说相关证书此时还没有完全注销掉，我们还需要编辑一下，如下所示：

```
vim /etc/server.conf
```

添加内容如下所示：

```
crl-verify /usr/local/src/OpenVPN-2.0.9/easy-rsa/2.0/keys/crl.pem
```

然后重新启动 OpenVPN，你会发现 andrewy 客户端已经不能登录了。

7.6.2 OpenVPN 服务器的安全问题

OpenVPN 作为企业内网的一个 VPN 入口，我们不建议作 DMZ 映射，其实只需要映射一个内网 IP，可以是 OpenVPN 的网桥的 IP 的 1194 的 TCP 和 UDP，这样做效果也非常好，不会发生丢包现象。

如果有条件的话，我们应该在防火墙多加一个网卡，增加一个 DMZ 区域，将 OpenVPN 服务器和要共享资源的服务器放在这个 DMZ 区域，跟我们的办公环境隔离开来，这样就算万一被黑客入侵了，给公司带来的损失也会减到最小。

下面说说关于用户证书的有效期限问题。

一般来说，我们在服务器端使用命令 `build-key *` 建立 Client 用户的时候，系统默认 Client 用户的 key 有效期限为 10 年。时间非常长，不符合公司的要求，因为公司是要求我们每 3 个月就换一次密码文件。如何更改这个有效期限呢？其实可以通过修改 `build-key` 文件来实现。我们可以编辑 `easy-rsa` 目录下的 `build-key` 文件，这是一个 SHELL 程序，将其默认的 3650 天改成自己需要的时间，如下所示：

```
if test $KEY_DIR;then
    cd $KEY_DIR &&\
    openssl req -days 3650 -nodes -new -keyout $1.key -out $1.csr -config $KEY_CONFIG &&\
```

```

    openssl ca -days 3650 -out $1.crt -in $1.csr -config $KEY_CONFIG && \
    chmod 0600 $1.key
else
    echo you must define KEY_DIR
fi

```

保存后再使用 `build-key *` 生成的 Client 用户的期限就会变成你想要的期限了，过期的 key 文件是登录不了服务器的。

7.6.3 OpenVPN 服务器的负载均衡

我们如何保证 OpenVPN 服务器的负载均衡呢？首先在不同的地方或不同 IP 的服务器上配置相同 OpenVPN 服务器，方法是先在其中的 A 服务器上配置好一个 VPN 服务器，把整个 OpenVPN 文件夹备份，然后在 B 服务器上安装一个 OpenVPN，再把之前 A 服务器上备份的 OpenVPN 文件夹里面的文件拷贝到 B 服务器上，覆盖掉原来的内容，这样两个服务器的配置就完全相同了，多个服务器则可以如法炮制（跟 DNS 轮询极类似）。

然后在客户端的配置文件中加上如下内容：

```

remote A 服务器 IP 端口
remote B 服务器 IP 端口
remote C 服务器 IP 端口
...
remote - random

```

此为随机选择服务器登录，要是 A 服务器登录不了或掉线了，会自动尝试登录 B 服务器，然后是 C 服务器，从而自动实现负载均衡。

在实际部署 OpenVPN Server 的备份时，我们觉得这办法很麻烦，所以单独配置了一台 PPTPD VPN 服务器，采用的是网通公网 IP，作为电信 OpenVPN 的备份服务器。如果同事们在家移动办公时，机房电信线路出问题，则可用 PPTPD VPN 联通服务器登录，这样就不会影响工作了。但有一个比较糟糕的情况是，PPTPD VPN 的穿透性很弱，一般的家用路由器都不支持 GRE，所以如果家里是用路由器 NAT 上网的话，一般连不上 PPTPD VPN 服务器，更谈不上连进公司的局域网环境了，所以大家还是喜欢连进 OpenVPN 办公。

7.7 小结

OpenVPN 是我现在非常喜欢的 VPN 服务器，它穿透性强，安全性和稳定性也非常之高。有越来越多的网络工程师也开始将其配置成 VPN 网关，用于 3 个机房（即双线机房、网通机房和电信机房）之间的 VPN 数据传输，这也是现在的流行趋势。我想大家通过本章的 OpenVPN 企业配置案例会很快熟悉 OpenVPN 的语法和工作模式，更多关于 OpenVPN 的细节问题希望大家在工作中挖掘，如果能更多地了解它的原理和细节，那么最终熟练掌握它并不是什么难事。



第 8 章 Linux 防火墙及系统安全

- 8.1 基础网络知识
- 8.2 Linux 防火墙的概念
- 8.3 Linux 防火墙在企业中的作用
- 8.4 Linux 防火墙的语法
- 8.5 iptables 的基础知识
- 8.6 如何流程化编写 iptables 脚本
- 8.7 学习 iptables 应该掌握的工具
- 8.8 iptables 的简单脚本学习
- 8.9 线上生产服务器的 iptables 脚本
- 8.10 TCP_wrappers 应用级防火墙的介绍和应用
- 8.11 工作中的 Linux 防火墙总结
- 8.12 Linux 系统自身的安全防护
- 8.13 Linux 系统安全相关的工具
- 8.14 Linux 服务器基础防护篇
- 8.15 如何防止入侵
- 8.16 小结

这一章我首先会向大家介绍 Linux 下的防火墙——Netfilter/iptables 的详细使用方法，在接下来的部分介绍 Linux 服务器下的安全防护手段。初学 Linux 防火墙的朋友可能会觉得 iptables 语法复杂，又是在纯字符下操作，不容易学习，其实我们只要掌握正确的方法，严格按照 iptables 的语法规则来执行，循序渐进，上手也是件很容易的事情。学习 iptables 跟学习英语一样，都是有语法和规律可循的，建议大家参考我所提供的 iptables 学习脚本和 iptables 线上脚本来学习，了解 iptables 的语法规则，相信很快就可以掌握 iptables 的用法了。

8.1 基础网络知识

这一节将向大家介绍几个关于网络的知识点，理解了这些基础点，将会方便你理解以后的 Linux 防火墙 iptables 的工作流程。

8.1.1 OSI 网络参考模型

OSI (Open System Interconnection) 模型表示一种层次型的网络架构。OSI 模型中的每一层都提供了关于其他层的独特功能。OSI 模型包括七层，如图 8-1 所示。

- 物理层：主要定义物理设备标准，如网线的接口类型、光线的接口类型、各种传输介质的传输速率等。它的主要作用是传输比特流（即由 1、0 转化为电流强弱来进行传输，到达目的地后再转化为 1、0，也就是我们常说的模数转换与数模转换）。这一层的数据叫做比特，网卡工作在此层。一般物理层较少关心网络入侵分析，而更关注于保证设备的电缆安全。
- 数据链路层：主要将从物理层接收的数据进行 MAC 地址（网卡的地址）的封装与解封装。我们常把这一层的数据叫做帧。在这一层工作的设备是交换机，数据通过交换机来传输（三层交换机不在此层工作）。
- 网络层：主要将从下层接收到的数据进行 IP 地址（例如 192.168.0.1/24）的封装与解封装。在这一层工作的设备是路由器，我们常把这一层的数据叫做数据包。
- 传输层：定义了一些传输数据的协议和端口号（如 www 端口 80 等），比如：TCP（传输控制协议，传输效率低，可靠性强，用于传输可靠性要求高且数据量大的数据）、UDP（用户数据报协议，与 TCP 的特性恰恰相反，用于传输可靠性要求不高且数据量小的数据，如 QQ 聊天数据就是通过这种方式传输的）。主要是将从下层接收的数据进行分段传输，到达目的地后再重组。我们常把这一层的数据叫做段。
- 会话层：通过传输层（端口号：传输端口与接收端口）建立数据传输的通路。主要在你的系统之间发起会话或接受会话请求（设备之间可通过 IP，也可通过 MAC 或主机名来相互认识）。
- 表示层：主要是对接收的数据进行解释、加密与解密、压缩与解压缩等（也就是把计算机能够识别的东西转换成人能够识别的东西，如图片、声音等）。



图 8-1 OSI 模型的七层参考模型

- 应用层：主要是一些终端的应用，比如说 FTP（各种文件下载）、Web（IE 浏览）、QQ 之类的（可把它理解成我们在计算机屏幕上可以看到的東西，也就是终端应用），也可以这样理解，应用层负责向用户或应用程序显示数据。

8.1.2 TCP/IP 三次握手/四次挥手的过程详解

1. 建立连接协议（三次握手）

- 1) 客户端发送一个带 SYN 标志的 TCP 报文到服务器。这是三次握手过程中的报文 1。
- 2) 服务器端回应客户端的是三次握手中的第 2 个报文。这个报文同时带 ACK 标志和 SYN 标志，它表示对刚才客户端 SYN 报文的回应，同时又将标志 SYN 给客户端，询问客户端是否准备好进行数据通信。
- 3) 客户端必须再次回应服务器端一个 ACK 报文，这是报文 3。

报文 (message) 是网络中交换与传输的数据单元。报文包含了将要发送的完整的数据信息，其长短很不一致。可分为自由报文和数字报文。报文也是网络传输的单位，传输过程中会不断地封装成分组、包、帧来传输，封装的方式就是添加一些信息段，那些是报文头。

2. 连接终止协议（四次挥手）

由于 TCP 连接是全双工的，因此每个方向都必须单独进行关闭。这原则是当一方完成它的数据发送任务时就发送一个 FIN 来终止这个方向的连接。收到一个 FIN 只意味着这一方向上没有数据流动了，一个 TCP 连接在收到一个 FIN 后仍能发送数据。首先关闭的一方将执行主动关闭，而另一方执行被动关闭。

- 1) TCP 客户端发送一个 FIN，用来关闭客户到服务器的数据传送（报文 4，一次挥手）。
- 2) 服务器收到这个 FIN，它发回一个 ACK，确认序号为收到的序号加 1（报文 5，二次挥手）。和 SYN 一样，一个 FIN 将占用一个序号。
- 3) 服务器关闭客户端的连接，发送一个 FIN 给客户端（报文 6，三次挥手）。
- 4) 客户端发回 ACK 报文确认，并将确认序号设置为收到的序号加 1（报文 7，四次挥手）。

在 TCP 三次握手/四次挥手的过程中存在着多种 TCP 连接状态，如下所示：

- CLOSED：这个没什么好说的了，表示初始状态。
- LISTEN：这也是非常容易理解的一个状态，表示服务器端的某个 SOCKET 处于监听状态，可以接受连接了。
- SYN_RCVD：这个状态表示接受到了 SYN 报文，在正常情况下，这个状态是服务器端的 SOCKET 在建立 TCP 连接时的三次握手会话过程中的一个中间状态，很短暂，基本上用 netstat 是很难看到这种状态的，除非你特意写了一个客户端测试程序，故意将三次 TCP 握手过程中最后一个 ACK 报文不予发送。在这种情况下，当收到客户端的 ACK 报文时，它会进入到 ESTABLISHED 状态。
- SYN_SENT：这个状态与 SYN_RCVD 遥相呼应，当客户端 SOCKET 执行 CONNECT 连接时，它首先会发送 SYN 报文，随即进入 SYN_SENT 状态，并等待服务端发送三次握手中的第 2 个报文。SYN_SENT 状态表示客户端已发送 SYN 报文。

- ESTABLISHED: 这个容易理解, 表示连接已经建立了。
- FIN_WAIT_1: 这个状态要好好解释一下, 其实 FIN_WAIT_1 和 FIN_WAIT_2 状态的真正含义都是表示等待对方的 FIN 报文。而这两种状态的区别是, FIN_WAIT_1 状态实际上是当 SOCKET 在 ESTABLISHED 状态时, 它想主动关闭连接, 于是向对方发送了 FIN 报文, 然后该 SOCKET 就进入到 FIN_WAIT_1 状态了。而在对方回应 ACK 报文后, 则进入 FIN_WAIT_2 状态, 当然在实际的正常情况下, 无论对方处于何种情况下, 都应该马上回应 ACK 报文, 所以 FIN_WAIT_1 状态一般是比较难见到的, 而 FIN_WAIT_2 状态还是常常可以用 netstat 看到的。
- FIN_WAIT_2: 上面已经详细解释了这种状态, 实际上 FIN_WAIT_2 状态下的 SOCKET, 表示半连接, 即有一方要求 close 连接, 同时还会告诉对方, 我暂时还有点数据需要传给你, 稍后再关闭连接。
- TIME_WAIT: 表示收到了对方的 FIN 报文, 并发送出了 ACK 报文, 就等 2MSL 后即可回到 CLOSED 可用状态。如果 FIN_WAIT_1 状态下, 收到了对方同时带 FIN 标志和 ACK 标志的报文, 可以直接进入到 TIME_WAIT 状态, 而无需经过 FIN_WAIT_2 状态。
- CLOSING: 这种状态比较特殊, 在实际应用中应该很少见, 属于一种比较罕见的例外状态。正常情况下, 当你发送 FIN 报文时, 按理来说是应该先收到 (或同时收到) 对方的 ACK 报文, 再收到对方的 FIN 报文。但是 CLOSING 状态表示你发送 FIN 报文后, 并没有收到对方的 ACK 报文, 而是收到了对方的 FIN 报文。在什么情况下会出现此状况呢? 其实细想一下, 也不难得出结论: 如果双方几乎在同时 close 一个 SOCKET 的话, 那么就会出现双方同时发送 FIN 报文的情况, 也即会出现 CLOSING 状态, 表示双方都正在关闭 SOCKET 连接。
- CLOSE_WAIT: 这种状态的含义其实是表示在等待关闭。怎么理解呢? 当对方 close 一个 SOCKET 后发送 FIN 报文给自己, 系统毫无疑问地会回应一个 ACK 报文给对方, 此时则进入到 CLOSE_WAIT 状态。实际上接下来你真正需要考虑的事情是, 查看你是否还有数据要发送给对方, 如果没有的话, 那么你也就可以 close 这个 SOCKET, 发送 FIN 报文给对方, 也即关闭连接。所以你在 CLOSE_WAIT 状态下, 需要完成的事情是等待, 然后关闭连接。
- LAST_ACK: 这个状态还是比较好理解的, 它是被动关闭一方在发送 FIN 报文后, 最后等待对方的 ACK 报文。当收到 ACK 报文时, 就表示可以进入 CLOSED 可用状态了。

提一个问题: 为什么 TIME_WAIT 状态还需要等 2MSL 后才能返回到 CLOSED 状态?

这是因为: 虽然双方都同意关闭连接了, 而且握手的 4 个报文也都协调和发送完毕, 按理可以直接回到 CLOSED 状态 (就好比从 SYN_SEND 状态到 ESTABLISHED 状态), 但是因为我们必须要假想网络是不可靠的, 你无法保证你最后发送的 ACK 报文一定会被对方收到, 比如对方处于 LAST_ACK 状态下的 SOCKET 可能会因为超时未收到 ACK 报文, 这时就需要重发 FIN 报文, 所以这个 TIME_WAIT 状态的作用就是用来重发可能丢失的 ACK 报文。

如图 8-2 所示的流程图能够帮助大家理解这个过程。

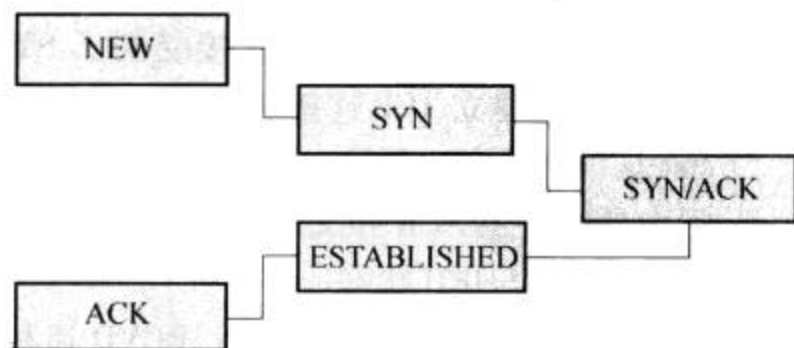


图 8-2 TCP/IP 三次握手流程图
(顺序为 NEW→SYN/ACK→ACK)

8.1.3 其他基础网络知识

其他关于网络的基础知识我也希望大家能够掌握，比如简单的子网划分、UDP 的连接原理、硬件防火墙的工作模式、SNAT 和 DNAT 等。如果确实对网络这块不熟悉，建议先预习下 CCNA 的基础知识和概念，这对于我们掌握接下来的 Linux 防火墙知识还是很有帮助的。无论你从事的是系统工作还是开发工作，基础的网络知识都是必不可少的，推荐大家有时间阅读一下 CCNA 的基础网络知识，本书限于篇幅就不做详细说明了。

8.2 Linux 防火墙的概念

Linux 防火墙其实并不是特别专业的叫法，我们所说的 Linux 防火墙其实指的是 Linux 下的 Netfilter/Iptables。Netfilter/Iptables 是与最新的 2.4.x/2.6.x 版本 Linux 核集成的 IP 信息包过滤系统。

虽然 Netfilter/Iptables IP 信息包过滤系统可作为一个整体来看，但其实它们是该过滤系统的两个组件，Netfilter 是内核的模块实现，Iptables 是上层的操作工具。Netfilter 是 Linux 核心中的一个通用架构，运行在内核空间（kernel space）。Iptables 提供一系列的表（tables），每个表由若干链（chains）组成，而每条链中可以由一条或数条规则（rule）组成，其规则则由一些信息包过滤表组成，这些表包含内核用来控制信息包过滤处理的规则集。Iptables 是一个管理内核包过滤的工具，可以加入、插入或删除核心包过滤表格中的规则。它运行在用户空间（user space）中，现在的发行版本中默认都有安装，但如果是在一些精简的系统上则可能无此工具，需要单独安装。实际上真正执行这些过滤规则的是 Netfilter。相对于 2.2 内核提供的 IP 链来说，iptables 实现的不仅仅是包过滤功能，而是通过 Netfilter 实现了一整套框架结构，在这个框架之上则实现了包过滤、NAT 等模块功能，从而提供了更好的可扩展性和灵活性。

Netfilter/iptables 的最大优点是它可以配置有状态的防火墙，这是 ipfwadm 和 ipchains 等以前的工具都无法提供的一种重要功能。有状态的防火墙能够指定并记住为发送或接收信息包所建立的连接状态。防火墙可以从信息包的连接跟踪状态获得该信息。在决定过滤新的信息包时，防火墙所使用的这些状态信息可以增加其效率和速度。这里有 4 种有效状态，其名称分别为 ESTABLISHED、INVALID、NEW 和 RELATED。状态 ESTABLISHED 指出该信息包属于已建立的连接，该连接一直用于发送和接收信息包并且完全有效。INVALID 状态指出该信息包与任何已知的流或连接都不相关联，它可能包含错误的数据或头。状态 NEW 意味着该信息包已经或将启动新的连接，或者它与尚未用于发送和接收信息包的连接相关联。最后，RELATED 表示该信息包正在启动新连接，以及它与已建立的连接相关联。由于 iptables 的状态我们在 iptables 脚本里用得比较多，在后面的章节中将会详细介绍说明之。

Netfilter/iptables 之前还有 ipfwadm 和 ipchains 程序，由于 iptables 跟这两个程序有很大的不同，所以本书将不会介绍它们。Netfilter/iptables 的另一个重要优点是，它使用户可以完全控制防火墙配置和信息包过滤。您可以定制自己的规则来满足您的特定需求，从而只允许您想要的网络流量进入系统。

另外，Netfilter/iptables 是免费的，这对于那些想要节省费用的人来说十分理想，它可以代替昂贵的防火墙解决方案。附带说明一下，iptables 和 Netfilter 的确存在差别，尽管它们经常被用来相互替换使用，Netfilter 是用来实现 Linux 内核中防火墙的 Linux 内核空间程序代码段，它要么被直接编译进内核，要么被包含在模块中。而 iptables 是用来管理 Netfilter 防火墙的用户程序，本书中，

iptables 包括 Netfilter 和 iptables，很多企业招聘面试的题目中有“区别 Netfilter 和 iptables 的概念”这样一题，希望大家也注意区别这二者的差别。

8.3 Linux 防火墙在企业中的作用

Linux 防火墙（即 Netfilter/Iptables IP 过滤系统）在企业中还是应用得非常广泛的。那么，它究竟应用在哪些方面呢？

1) 很多中小企业和网吧利用 iptables 来作企业的 NAT 路由器，代替传统的路由器供企业内部员工上网。在节约成本和有效控制上，Linux 防火墙确实有它独有的优势，像武汉流行的海蜘蛛路由器，其实就相当于 iptables 的二次开发（在其基础上增加了许多中文模块）。

2) IDC 机房的服务器可以用 Linux 防火墙代替硬件防火墙作为主机的防护措施，由于 IDC 机房的机器一般是没有硬件防火墙的，用开源免费的 iptables 是一个性价比比较高的选择。

3) iptables 可以结合 squid 作为企业内部上网的透明代理。传统的代理需要在浏览器里配置代理服务器信息，而 iptables + squid 的透明代理则可以把客户端的请求重定向到代理服务器的端口，让客户端感觉不到代理的存在，当然，客户端也无需做任何代理设置。

4) 当将 iptables 作为企业 NAT 路由器时，我们可以使用 iptables 的扩展模块屏蔽 P2P 流量，还可以禁止非法网页。这个功能可以供有需求的朋友使用，不过，我个人建议还是让 iptables 发挥最大功效，非法流量完全可以通过行政手段来控制。

5) iptables 还可以用于外网 IP 向内网 IP 映射。我们可以假设有一家 ISP 提供园区 Internet 接入服务，为了方便管理，该 ISP 分配给园区用户的 IP 地址都是内网 IP，但是部分用户要求建立自己的 Web 服务器对外发布信息。我们可以在防火墙的外部网卡上绑定多个合法 IP 地址，然后通过 IP 映射使发给其中某一个 IP 地址的包转发至内部用户的 Web 服务器上，这样内部的 Web 服务器也就可以对外提供服务了。这种形式的 NAT 一般称为 DNAT，在前面的集群架构的环节，我们经常将负载均衡器的内网 VIP 的 80 和 443 端口通过防火墙映射成公网 IP 的 80 和 443 端口，这也是 DNAT 的完美实现。

6) iptables 可以防止轻量级的 DOS 攻击，比如 ping 攻击及 SYN 洪水攻击，我们利用 iptables 来做相关安全策略还是很有效果的。

8.4 Linux 防火墙的语法

对于数据包而言，也有以下几个流向：

PREROUTING→FORWARD→POSTROUTING

PREROUTING→INPUT→本机 OUTPUT→POSTROUTING

大家可以留意下，数据包的两种主要流向其实也是我们后面 iptables 的两种工作模式：一是作为 NAT 路由器，另一种是作为主机防火墙，所以我们对应地要在 iptables 的规则链上做文章。iptables 数据流入和流出详细流程如图 8-3 所示。

iptables 根据不同的数据处理包处理功能使用不同的规则表。它包括如下三个表：filter、nat 和 mangle。

filter 是默认的表，它包含真正的防火墙过滤规

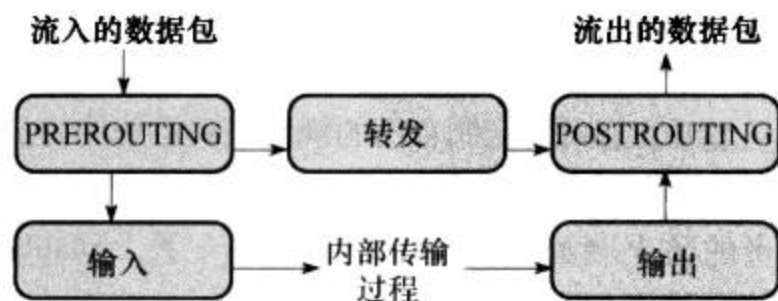


图 8-3 iptables 数据包流入和流出详细流程图

则。内建的规则链包括：INPUT、OUTPUT 和 FORWARD。

nat 表包含源和目的地址及端口转换使用的规则，内建的规则链包括 PREROUTING、OUTPUT 和 POSTROUTING。

mangle 表包含用于设置特殊的数据包路由标志的规则。这些标志随后被 filter 表中的规则检查。内建的规则链包括：PREROUTING、INPUT、FORWARD、POSTROUTING 和 OUTPUT。

表对应的相关规则链的功能如下。

- INPUT 链：当一个数据包由内核中的路由计算确定为本地的 Linux 系统后，它会通过 INPUT 链的检查。
- OUTPUT 链：保留给系统自身生成的数据包。
- FORWARD 链：经过 Linux 系统路由的数据包（即当 iptables 防火墙用于连接两个网络时，两个网络之间的数据包必须流经该防火墙）。
- PREROUTING 链：用于修改目标地址（DNAT）。
- POSTROUTING 链：用于修改源地址（SNAT）。

iptables 详细语法如下所示：

```
iptables [-t 表名] [-A | I | D | R] 链名 [规则编号] [-i | o 网卡名称] [-p 协议类型] [-s 源 IP 地址 | 源子网] [--sport 源端口号] [-d 目标 IP 地址 | 目标子网] [--dport 目标端口号] [-j 动作]
```

注意 此语法规则详细，逻辑清晰，推荐以此语法记忆。我们在刚开始写 iptables 规则时就应该养成好习惯，用公式来规范脚本，这对于我们以后的工作大有帮助。

下面是关于语法的详细说明。

(1) 定义默认策略。

作用：当数据包不符合链中任一条规则时，iptables 将根据该链预先定义的默认策略来处理数据包。

默认策略的定义格式为：

```
iptables [-t 表名] [-P] 链名 动作
```

参数说明如下：

[-t 表名]

指默认策略将应用于哪个表，可以使用 filter、nat 和 mangle。如果没有指定使用哪个表，iptables 就默认使用 filter 表。

<-P>

定义默认策略。

<链名>

指默认策略将应用于哪个链，可以使用 INPUT、OUTPUT、FORWARD、PREROUTING、OUTPUT 和 POSTROUTING。

<动作>

处理数据包的动作，可以使用 ACCEPT（接收数据包）和 DROP（丢弃数据包）。

（2）查看 iptables 规则

查看 iptables 规则的命令格式为：

```
iptables [-t 表名] <-L> [链名]
```

参数说明如下：

[-t 表名]

指查看哪个表的规则列表，表名用可以使用 filter、nat 和 mangle。如果没有指定使用哪个表，iptables 就默认查看 filter 表的规则列表。

<-L>

查看指定表和指定链的规则列表。

[链名]

指查看指定表中哪个链的规则列表，可以使用 INPUT、OUTPUT、FORWARD、PREROUTING、OUTPUT 和 POSTROUTING。如果不指明哪个链，则将查看某个表中所有链的规则列表。

（3）增加、插入、删除、替换 iptables 规则

参数说明如下：

[-t 表名]

定义默认策略将应用于哪个表，可以使用 filter、nat 和 mangle。如果没有指定使用哪个表，iptables 就默认使用 filter 表。

-A

新增加一条规则，该规则将会增加到规则列表的最后一行，该参数不能使用规则编号。

-I

插入一条规则，原本该位置上的规则将会往后顺序移动，如果没有指定规则编号，则在第一条规则前插入。

-D

从规则列表中删除一条规则，可以输入完整规则，或直接指定规则编号加以删除。

-R

替换某条规则，必须要指定替换的规则编号，规则被替换并不会改变顺序。

<链名>

指定查看指定表中哪个链的规则列表，可以使用 INPUT、OUTPUT、FORWARD、PREROUTING、OUTPUT 和 POSTROUTING。

[规则编号]

规则编号在插入、删除和替换规则时用，编号是按照规则列表的顺序排列的，规则列表中第一条规则的编号为 1。

`[-i | o 网卡名称]`

i 是指定数据包从哪块网卡进入, o 是指定数据包从哪块网卡输出。网卡名称可以使用 ppp0、eth0 和 eth1 等。

`[-p 协议类型]`

可以指定规则应用的协议, 包含 TCP、UDP 和 ICMP 等。

`[-s 源 IP 地址 | 源子网]`

源主机的 IP 地址或子网地址。

`[-s sport 源端口号]`

数据包的 IP 源端口号。

`[-d 目标 IP 地址 | 目标子网]`

目标主机的 IP 地址或子网地址。

`[-d dport 目标端口号]`

数据包的 IP 目标端口号。

`<-j 动作>`

下面是处理数据包的动作以及各个动作的详细说明。

- ACCEPT: 接收数据包。
- DROP: 丢弃数据包。
- REDIRECT: 将数据包重新转向到本机或另一台主机的某一个端口, 通常能实现透明代理或对外开放内网的某些服务。
- REJECT: 拦截该数据封包, 并发回封包通知对方。
- SNAT: 源地址转换, 即改变数据包的源地址。例如: 将局域网的 IP (10.0.0.1/24) 转换为广域网的 IP (203.93.236.141/24), 在 NAT 表的 POSTROUTING 链上进行该动作。
- DNAT: 目标地址转换, 即改变数据包的目的地址。例如: 将广域网的 IP (203.93.236.141/24) 转换为局域网的 IP (10.0.0.1/24), 在 NAT 表的 PREROUTING 链上进行该动作。
- MASQUERADE: IP 伪装, 即常说的 NAT 技术, MASQUERADE 只能用于 ADSL 等拨号上网的 IP 伪装, 也就是主机的 IP 是由 ISP 分配动态的, 如果主机的 IP 地址是静态固定的, 就要使用 SNAT。
- LOG: 日志功能, 将符合规则的数据包相关信息记录在日志中, 以便管理员的分析和排错。

(4) 清除规则和计数器

在新建规则时, 往往需要清除原有的、旧的规则, 以免它们影响新设定的规则。如果规则比较多, 一条条删除就会十分麻烦, 这时可以使用 iptables 提供的清除规则参数达到快速删除所有的规则的目的。

定义参数的格式为:

`iptables [-t 表名] <-F | Z>`

参数说明如下：

`[-t 表名]`：指定默认策略将应用于哪个表，可以使用 `filter`、`nat` 和 `mangle`。如果没有指定使用哪个表，`iptables` 就默认使用 `filter` 表。

`-F`：删除指定表中的所有规则。

`-Z`：将指定表中的数据包计数器和流量计数器归零。

8.5 iptables 的基础知识

8.5.1 iptables 的状态 state

在 8.2 节里提到了 `state` 这个定义，这里将解释一下 `iptables` 防火墙的状态（`state`）。

比如，当我们用 `PieTTY` 远程访问远程主机的 `SSH` 端口时，我们的主机会和远程主机进行通信。此时，静态的防火墙会如下处理：

检查进入机器的数据包，发现数据的来源是 22 端口，当这些数据包允许时进入，连接之后相互通信的数据也一样，检查每个数据，如果发现数据包来源于 22 端口，允许通过。

如果用有状态的防火墙如何处理呢？

在你连接远程主机成功之后，你的主机会把这个连接记录下来，当有数据从远程 `SSH` 服务器进入你的机器时，它会检查自己的连接状态表，若发现这个数据来源于一个已经建立的连接，则允许这个数据包进入。

以上两种处理方法，很明显静态防火墙比较生硬，而 `iptables` 防火墙相对智能一些，这也是 `iptables` 防火墙的特点之一。

下面将解释几种状态：

- `NEW`：如果你的主机向远程机器发出一个连接请求，这个数据包的状态是 `NEW`。
- `ESTABLISHED`：在连接建立之后（完成 `TCP` 的三次握手后），远程主机和你的主机通信数据的状态为 `ESTABLISHED`。
- `RELATED`：和现有联机相关的新联机封包。像 `FTP` 这样的服务，用 21 端口传送命令，而用 20 端口（`port` 模式）或其他端口（`PASV` 模式）传送数据。在已有的 21 端口上建立好连接后发送命令，用 20 或其他端口传送的数据（`FTP-DATA`），其状态是 `RELATED`。
- `INVALID`：无效的数据包，不能被识别属于哪个连接或没有任何状态，通常这种状态的数据包会被丢弃。

有了以上知识后，接下来我们可以进行一个简单的实验了。

首先，我们还是来设置默认规则，如下所示：

```
iptables -F INPUT DROP
```

这样你的机器会将所有进入你主机的数据都丢弃掉。

如果你有一台主机只是用于个人桌面应用的，也就是此主机不提供任何服务，那么，我们就禁止其他的机器向你的机器发送任何连接请求，命令如下：

```
iptables -A INPUT -m state --state NEW -j DROP
```

这个规则将所有发送到你机器的数据包（状态是 `NEW` 的包）丢弃。也就是不允许其他的机器

主动发起对你机器的连接，但是你却可以主动地连接其他的机器，不过仅仅是连接而已。连接之后的数据就是 ESTABLISHED 状态的了。我们再加上下面这一条语句，这一条语句的作用是允许所有已经建立连接或者与之相关的数据通过，如下所示：

```
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

好了，我们将上面的语句归纳一下，可以来写一个简单的 iptables 脚本，作为个人桌面主机的防火墙，如下所示：

```
#!/bin/bash
iptables -F
iptables -F -t nat
iptables -X
iptables -Z
iptables -P INPUT DROP
iptables -A INPUT -m state --state NEW -j DROP
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

给此脚本 x 权限，`chmod +x iptables.sh`（名字随便定义即可），让其执行即可。

前面几条语句是将其默认规则全部清除，让此脚本后面的语句生效。

其实，第二条可以被注释掉了，那一规则完全可以省去，让默认规则处理就行了。

是不是很简单呢？对于个人桌面应用来说，只用刚才那两条语句，就能让你接入 Internet 网的主机足够安全。而且你可以随意访问 Internet，但是外部却不能主动发起对你机器的连接。

我们可以看到有状态的防火墙比静态防火墙要“智能”一些，当然规则也容易设置一些。

执行以上脚本后，我们可以查看一下 iptables 规则，命令如下：

```
iptables -nv -L
```

命令显示结果如下：

```
Chain INPUT (policy DROP 0 packets, 0 bytes)
  pkts bytes target    prot opt in     out     source            destination
    0    0 DROP      all  --  *      *        0.0.0.0/0         0.0.0.0/0         state NEW
   37 2520 ACCEPT    all  --  *      *        0.0.0.0/0         0.0.0.0/0         state
RELATED,ESTABLISHED
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target    prot opt in     out     source            destination
Chain OUTPUT (policy ACCEPT 1532 packets, 1224K bytes)
  pkts bytes target    prot opt in     out     source            destination
```

另外，在执行以上脚本后，我们会发现此机器会拒绝一切数据接入，但是我们原先的 SSH 并没有被断开。这是为什么呢？这就是因为 `iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT` 这句话发挥了作用，我们原先建立的连接还存在，所以主机不会将此 ESTABLISHED 连接断掉，state 的优势在这里发挥得淋漓尽致。

注意 以上脚本在实验环境下尝试即可，切勿应用于生产服务器，因为默认会拒绝一切连接的。

参考资料：<http://197962.blog.chinajavaworld.com/entry/480/0/>

8.5.2 iptables 的 Conntrack 记录

我们先来看看怎样阅读 `/proc/net/ip_conntrack` 里的 conntrack 记录。这些记录表示的是当前被跟踪的连接。如果安装了 `ip_conntrack` 模块，我们可以查看 `ip_conntrack` 记录，命令如下：

```
cat /proc/net/ip_conntrack
tcp      6 117 SYN_SENT src=192.168.1.6 dst=192.168.1.9 sport=32775 dport=22 [UNREPLIED]
src=192.168.1.9 dst=192.168.1.6 sport=22 dport=32775 use=2
tcp      6 431967 ESTABLISHED src=192.168.18.120 dst=192.168.18.98 sport=1059 dport=22
packets=3287 bytes=262840 src=192.168.18.98 dst=192.168.18.120 sport=22 dport=1059
packets=2863 bytes=439784 [ASSURED] mark=0 secmark=0 use=
```

Conntrack 模块维护的所有信息都包含在这个例子中了，通过它们就可以知道某个特定的连接处于什么状态。首先显示的是协议，这里是 `tcp`，接着是十进制的 6（`tcp` 的协议类型代码是 6）。之后的 117 是这条 conntrack 记录的生存时间，它会有规律地被消耗，直到收到这个连接的更多的包。那时，这个值就会被设为当时那个状态的默认值。接下来的是这个连接在当前时间点的状态。上面的例子说明这个包处在状态 `SYN_SENT`，这个值是 iptables 显示的，以便我们好理解，而内部用的值稍有不同。`SYN_SENT` 说明我们正在观察的这个连接只在一个方向发送了一个 TCP SYN 包。再下面是源地址、目的地址、源端口和目的端口。其中有个特殊的词 `UNREPLIED`，说明这个连接还没有收到任何回应。最后，是希望接收的应答包的信息，它们的地址和端口和前面是相反的。

连接跟踪记录的信息依据 IP 所包含的协议不同而不同，所有相应的值都是在头文件 `linux/include/netfilter-ipv4/ip_conntrack *.h` 中定义的。IP、TCP、UDP、ICMP 协议的默认值是在 `linux/include/netfilter-ipv4/ip_conntrack.h` 里定义的。具体的值可以查看相应的协议，但我们这里用不到它们，因为它们大都只在 conntrack 内部使用。随着状态的改变，生存时间也会改变。

当一个连接在两个方向上都有传输时，conntrack 记录就删除 `[UNREPLIED]` 标志，然后重置。在末尾有 `[ASSURED]` 的记录说明两个方向已没有流量。这样的记录是确定的，在连接跟踪表满时，是不会被删除的，没有 `[ASSURED]` 的记录就要被删除。连接跟踪表能容纳多少记录是被一个变量控制的，它可由内核中的 `ip-sysctl` 函数设置。默认值取决于你的内存大小，128MB 可以包含 8192 条目录，256MB 是 16 376 条，大家如果在生产服务器上通过加载模块的方法开启了 `ip_conntrack` 功能，就要注意内存方面的使用情况，此模块是极消耗内存的，对系统非常有影响，所以我这里不建议在生产服务器上开启 conntrack 功能。也可以在 `/proc/sys/net/ipv4/ip_conntrack_max` 里查看、设置。

8.5.3 关于 iptables 模块的说明

大多数 Linux 版本实现 iptables 时使用一系列可载入的程序模块，几乎所有的模块在第一次使用时都被动态地自动载入。当然了，我们在撰写 iptables 脚本时也可以有选择地载入模块。现在许多朋友喜欢自己开发新的模块来实现更为强大的功能，这个问题不是本书重点，有兴趣的朋友可以自行开发和测试。

8.5.4 iptables 防火墙初始化的注意事项

在跟一些系统管理员进行交流时，我发现大家经常遇到的一个问题就是：误操作 iptables 而将

自己也拦截在机器之外了，没办法只有去机房重启 iptables。其实这个问题是有办法解决的，特推荐给大家，建议大家学习参考。

我们可以配置一计划任务 Crontab，每 5 分钟运行一次，即 `*/5 */* */* /etc/init.d/iptables stop;`，这样即使你的脚本存在错误设置（或丢失的）规则时，也不至于将你锁在计算机外而无法返回与计算机的连接，可让你放心大胆地调试你的脚本。鉴于许多同事在学习及调试 iptables 脚本时也是用自己内部的机房，所以推荐用此方法，免得受进出机房之苦。

8.5.5 如何保存运行中的 iptables 规则

使用 iptables-save 和 iptables-restore 的一个最重要的原因是，它们能在相当程度上提高装载、保存规则的速度。使用脚本更改规则的一个问题是，改动每个规则都要调用命令 iptables，而每一次调用 iptables，它首先要将 Netfilter 内核空间中的整个规则集都提取出来，然后再插入或附加，或做其他的改动，最后，再把新的规则集从它的内存空间插入到内核空间中，这会花费很多时间。

为了解决这个问题，可以使用命令 iptables-save 和 iptables-restore。iptables-save 用来把规则集保存到一个特殊格式的文本文件里，而 iptables-restore 是用来把这个文件重新装入内核空间的。这两个命令最好的地方在于一次调用就可以装载和保存规则集，而不像在脚本中每个规则改动都要调用一次 iptables。iptables-save 运行一次就可以把整个规则集从内核里提取出来，并保存到文件里，而 iptables-restore 每次装入一个规则表。换句话说，对于一个很大的规则集，如果用脚本来设置，那这些规则就会反反复复地被卸载、安装，而我们现在可以把整个规则集一次就保存下来，安装时也只是一次一个表，这可节省大量的时间。如果你的工作对象是一组巨大的规则，采用这两个工具将是明智的选择。

系统启动 iptables 的规则后，默认就有如下规则（虽然比较人性化，但很多时候达不到我们的要求，所以这里大家了解一下就好，不需要作深入研究，在这里我们可以用 cat 命令来查看）：

```
cat /etc/sysconfig/iptables
# Firewall configuration written by system-config-securitylevel
# Manual customization of this file is not recommended.
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:RH-Firewall-1-INPUT - [0:0]
-A INPUT -j RH-Firewall-1-INPUT
-A FORWARD -j RH-Firewall-1-INPUT
-A RH-Firewall-1-INPUT -i lo -j ACCEPT
-A RH-Firewall-1-INPUT -p icmp --icmp-type any -j ACCEPT
-A RH-Firewall-1-INPUT -p 50 -j ACCEPT
-A RH-Firewall-1-INPUT -p 51 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport 5353 -d 224.0.0.251 -j ACCEPT
-A RH-Firewall-1-INPUT -p udp -m udp --dport 631 -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp -m tcp --dport 631 -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A RH-Firewall-1-INPUT -j REJECT --reject-with icmp-host-prohibited
COMMIT
```


目前编写和调试 iptables 防火墙的通用做法还是通过脚本来进行，这相对而言更为方便，调试的效率也更高，特别是当我们按照标准流程来编写 iptables 脚本以后。当然这两种方法各有各的优点，我们可以根据自己的环境来选择到底用哪种方法来保存我们的 iptables 脚本。我个人还是倾向于自己手动编写 iptables 脚本。

8.6 如何流程化编写 iptables 脚本

1) 根据需求调整系统内核，例如 TCP 的 SYN 缓冲（cookies）是一种快速检测和防御 SYN 洪水攻击的机制，如下的命令可以启用 SYN 缓冲：

```
echo "1" > /proc/sys/net/ipv4/tcp_syncookies
```

另外，如果有 iptables 作 NAT 路由器，如果存在着多网卡的情况，这个时候要开启 IP 转发功能，用于多网卡之间数据的流通，命令如下：

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

其他适用于 iptables 防火墙的内核调整大家可以根据需求自行设定。

2) 加载 iptables 模块，由于我们这里不是以服务的方式启动 iptables，即/etc/init.d/iptables start，所以需要手动加载 iptables 模块，例如：

```
modprobe ip_tables
modprobe iptable_nat
modprobe ip_nat_ftp
modprobe ip_nat_irc
modprobe ip_conntrack
modprobe ip_conntrack_ftp
modprobe ip_conntrack_irc
modprobe ipt_MASQUERADE
```

接下来我们可以用 lsmod 来查看下加载的模块，命令如下：

```
lsmod | grep "ip_"
```

此命令显示结果如下：

```
ip_nat_irc          35777  0
ip_conntrack_irc    40465  1 ip_nat_irc
ip_nat_ftp          36545  0
ip_conntrack_ftp    41361  1 ip_nat_ftp
ip_nat              52973  4 ipt_MASQUERADE,ip_nat_irc,ip_nat_ftp,iptable_nat
ip_conntrack_netbios_ns  36033  0
ip_conntrack        91621  9 ipt_MASQUERADE,ip_nat_irc,ip_conntrack_irc,ip_nat_ftp,ip_
conntrack_ftp,iptable_nat,ip_nat,ip_conntrack_netbios_ns,xt_state
nfnetlink           40457  2 ip_nat,ip_conntrack
ip_tables           55329  2 iptable_nat,iptable_filter
x_tables            50505  7 ipt_MASQUERADE,iptable_nat,ipt_REJECT,xt_state,xt_tcpudp,ip_
tables,ip6_tables
```

3) 清空所有的表链规则（包括自定义的），命令如下：


```
iptables -F
iptables -X
iptables -Z
iptables -F -t nat
iptables -X -t nat
iptables -Z -t nat
iptables -X -t mangle
```

4) 定义默认策略，一般说来为了搭建安全的防火墙，默认是拒绝一切的，所以三表五链默认都应该是 DROP，但在实际线上的 iptables 脚本，我会建议如下配置（因为从服务器 OUTPUT 出去的数据和 NAT 出去的数据一般认为是安全的）：

```
iptables -P INPUT DROP
iptables -P FORWARD ACCEPT
iptables -P OUTPUT ACCEPT
iptables -t nat -P PREROUTING ACCEPT
iptables -t nat -P POSTROUTING ACCEPT
iptables -t nat -P OUTPUT ACCEPT
```

5) 打开“回环”，以免不必要的麻烦。

```
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
```

6) 允许状态为 ESTABLISHED 状态的数据包。

```
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

7) 根据需求建立防火墙规则，比如我们上一章节完整的桌面主机防火墙脚本如下所示：

```
#!/bin/bash
modprobe ip_tables
modprobe iptable_nat
modprobe ip_nat_ftp
modprobe ip_nat_irc
modprobe ip_conntrack
modprobe ip_conntrack_ftp
modprobe ip_conntrack_irc
modprobe ipt_MASQUERADE

iptables -F
iptables -X
iptables -Z
iptables -F -t nat
iptables -X -t nat
iptables -Z -t nat
iptables -X -t mangle

iptables -P INPUT DROP
iptables -P FORWARD ACCEPT
iptables -P OUTPUT ACCEPT
iptables -t nat -P PREROUTING ACCEPT
```

```
iptables -t nat -P POSTROUTING ACCEPT
iptables -t nat -P OUTPUT ACCEPT

iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

iptables -A INPUT -m state --state NEW -j DROP
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

此脚本一旦执行，本来正常提供的 Samba 服务马上断开了，客户端连接此机的 Samba 报错。我们可以查看 iptables 的 conntrack 记录，命令如下：

```
cat /proc/net/ip_conntrack
```

此命令显示结果如下：

```
udp      17 15 src=192.168.1.101 dst=192.168.1.255 sport=137 dport=137 packets=20 bytes=
1920 [UNREPLIED] src=192.168.1.255 dst=192.168.1.101 sport=137 dport=137 packets=0 bytes=0
mark=0 secmark=0 use=1
udp      17 12 src=192.168.1.101 dst=192.168.1.255 sport=138 dport=138 packets=1 bytes=269
[UNREPLIED] src=192.168.1.255 dst=192.168.1.101 sport=138 dport=138 packets=0 bytes=0 mark=0
secmark=0 use=1
```

Samba 建立连接的 137、138 端口只有一边有流量，TCP 的三次握手被拒，这个肯定是提供不了 Samba 服务的，证明此脚本是有效的。

8) 给脚本 x 权限，以后就可以直接执行此脚本了。

大家编写 iptables 脚本时可以参考上述步骤，一步步地来，这样不容易出错，以后熟练了就习惯成自然了。

8.7 学习 iptables 应该掌握的工具

8.7.1 命令行的抓包工具 TCPDump

TCPDump 是入侵分析人员工具包中的一个重要工具。在底层上，TCPDump 是一个捕获和分析数据包的软件，也就是说 TCPDump 可以用来监听网络通信，但是实际能够监听到什么样的数据流将取决于所在网络的拓扑结构。它是一款基于命令行的工具，可以通过不同的命令行选项来改变其状态，它也提供丰富的选项可使我们很容易地改变程序的运行方式。在使用 TCPDump 的过程中，我们会发现大部分捕获数据的动作只需要用到一些常用的选项，而不需要用到其他全部的选项。另外，TCPDump 存在于绝大多数的 Linux 系统和 FreeBSD 系统中，这是不需要安装即可使用的。下面就以具体例子来说明它的使用方法：

1) 想要截获所有 210.27.48.1 的主机收到的和发出的数据包，命令如下：

```
tcpdump host 210.27.48.1
```

2) 想要截获主机 210.27.48.1 和主机 210.27.48.2 或 210.27.48.3 的通信，使用如下命令（在命令行中使用括号时，要用转义符 \ 来对括号进行转义）：

```
tcpdump host 210.27.48.1 and \(210.27.48.2 or 210.27.48.3 \)
```

3) 如果想要获取主机 210.27.48.1 和所有主机 (除了 210.27.48.2 之外) 通信的 IP 包, 使用如下命令:

```
tcpdump ip host 210.27.48.1 and ! 210.27.48.2
```

4) 如果想要获取主机 210.27.48.1 接收或发出的 smtp 包, 使用如下命令:

```
tcpdump tcp port 25 and host 210.27.48.1
```

5) 如果怀疑系统正受到拒绝服务 (DoS) 攻击, 网络管理员可以通过截获发往本机的所有 ICMP 包, 来确定目前是否有大量的 ping 指令流向服务器, 此时可用如下命令:

```
tcpmdump icmp -n -i eth0
```

6) 如果想将其结果生成详细的报告, 可以用如下命令:

```
tcpdump tcp port 25 and host 211.147.1.11 > awstat.txt
```

用 TCPDump 捕获的 TCP 包的一般输出信息如下:

```
src > dst: flags data - seqno ack window urgent options
```

这里说明一下 TCPDump 抓取 TCP 包的情况。

src > dst: 表明从源地址到目的地址; flags 是 TCP 包中的标志信息, S 代表 SYN 标志, F 代表 FIN, P 代表 PUSH, R 代表 RST, “.” 表示没有标记; data-seqno 是数据包中数据的顺序号, ack 是下次期望的顺序号, window 是接收缓存的窗口大小, urgent 表明数据包中是否有紧急指针, options 是选项。

由于我们涉及的服务器协议大部分是 TCP 协议的, 这里只介绍了 TCP 包的输出信息。UDP 和 ICMP 协议信息大家可以根据上面介绍的 TCPDump 语法自行研究, 限于篇幅, 这里不做详细说明。

8.7.2 图形化抓包工具 Wireshark

Wireshark 是世界上最流行的网络分析工具。这个强大的工具可以捕捉网络中的数据, 并为用户提供关于网络和上层协议的各种信息。与很多其他网络工具一样, Wireshark 也使用 pcap network library 来进行封包捕捉, Wireshark 的前身是 Ethereal。网络管理员使用 Wireshark 来检测网络问题, 网络安全工程师使用 Wireshark 来检查资讯安全相关问题, 开发者使用 Wireshark 来为新的通信协定除错, 普通使用者使用 Wireshark 来学习网络协定的相关知识。Wireshark 不是入侵监测软件 (Intrusion Detection Software, IDS), 对于网络上的异常流量行为, Wireshark 不会产生警示或任何提示。然而, 仔细分析 Wireshark 捕捉的封包能够帮助使用者对于网络行为有更清楚的了解。Wireshark 不会对网络封包内容进行修改, 它只会反映出目前流通的封包信息。Wireshark 本身也不会把封包送至网络上。

Wireshark 在 Centos5.5 x86_64 下的安装极为方便。首先我们准备一台安装了图形化界面的机器 (因为 Wireshark 是基于图形化的工具), 执行如下命令:

```
yum -y install wireshark
```

然后在命令下面用 X-manager3.0 的 Xshell3.0 登录, 直接输入命令 wireshark 即可。

下面以一个简单的例子来说明一下，例如我的客户端是 192.168.1.100，我用 Xshell3.0 连到了 192.168.1.101 的 Server 上面，端口为 22，我们如何用 Wireshark 来进行抓包工作呢？步骤如下：

- 1) 在 Server 上面执行 Wireshark，打开此工具，如图 8-4 所示。

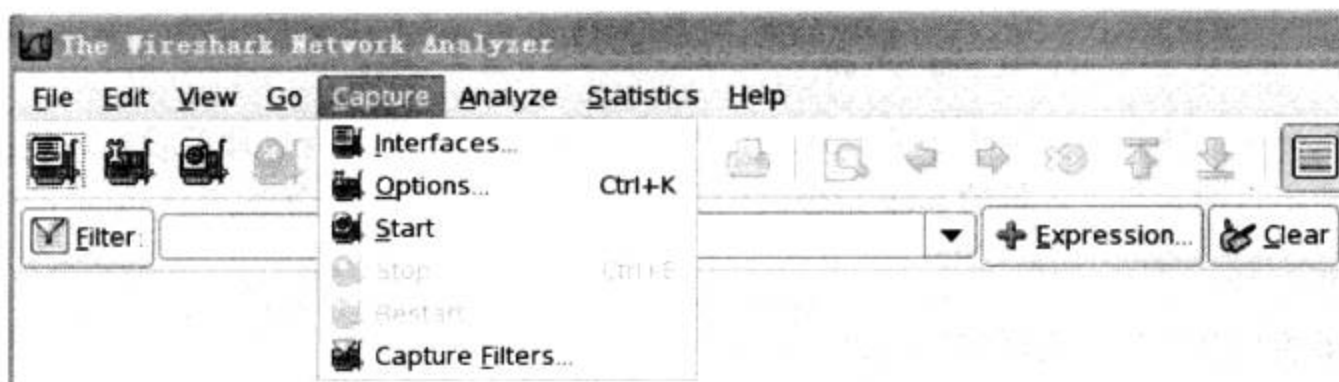


图 8-4 Wireshark 工作界面

- 2) 选择“Capture”的“options”选项，在“Capture Filters”里面输入我们的抓包规则如下：

host 192.168.1.100 and (host 192.168.1.101 and port 22)

Wireshark 语法跟 TCPDump 非常接近，如图 8-5 所示。

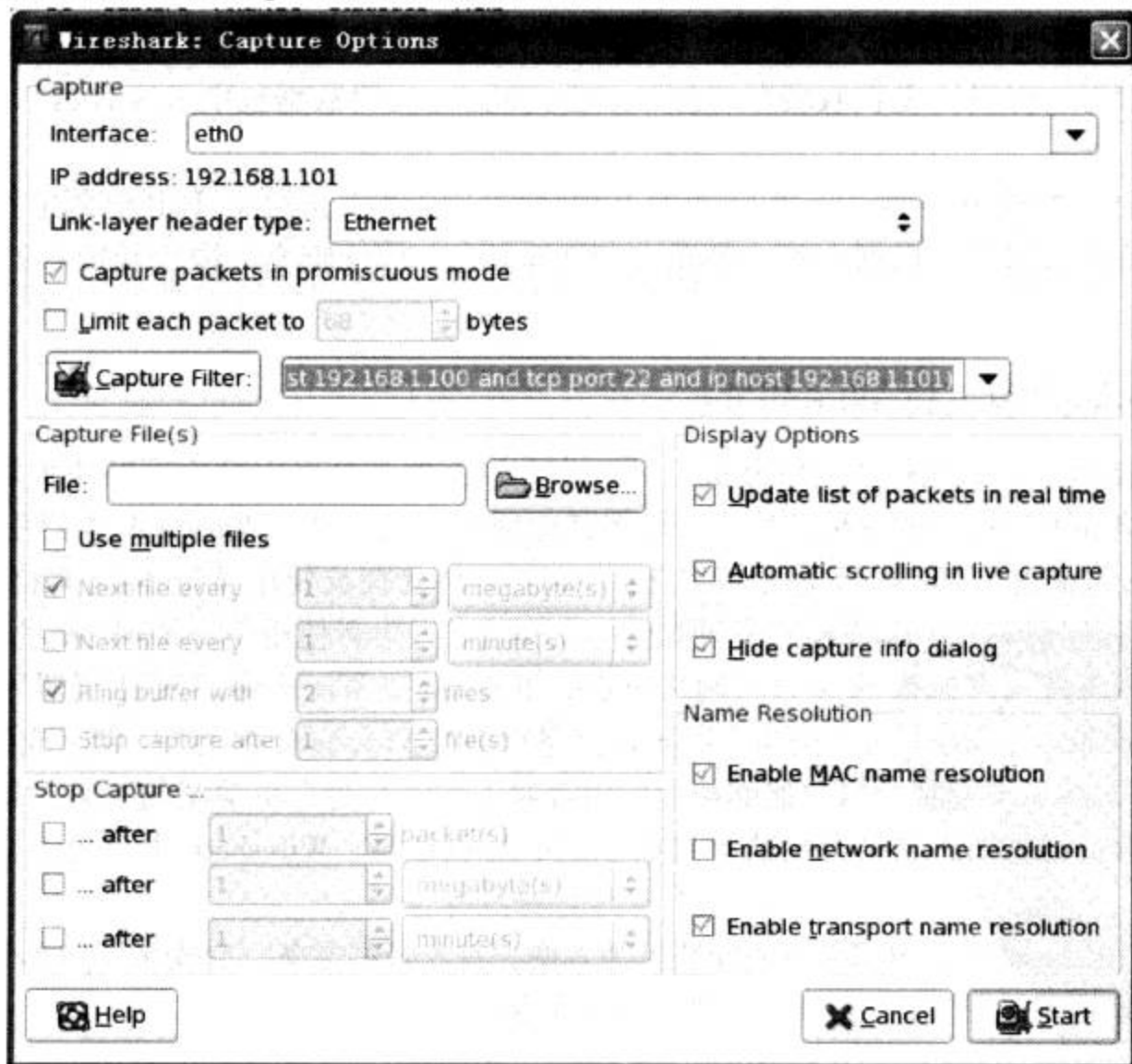


图 8-5 编写 Wireshark 抓包规则

3) 点击“Start”，就可以开始抓包，如图8-6所示。

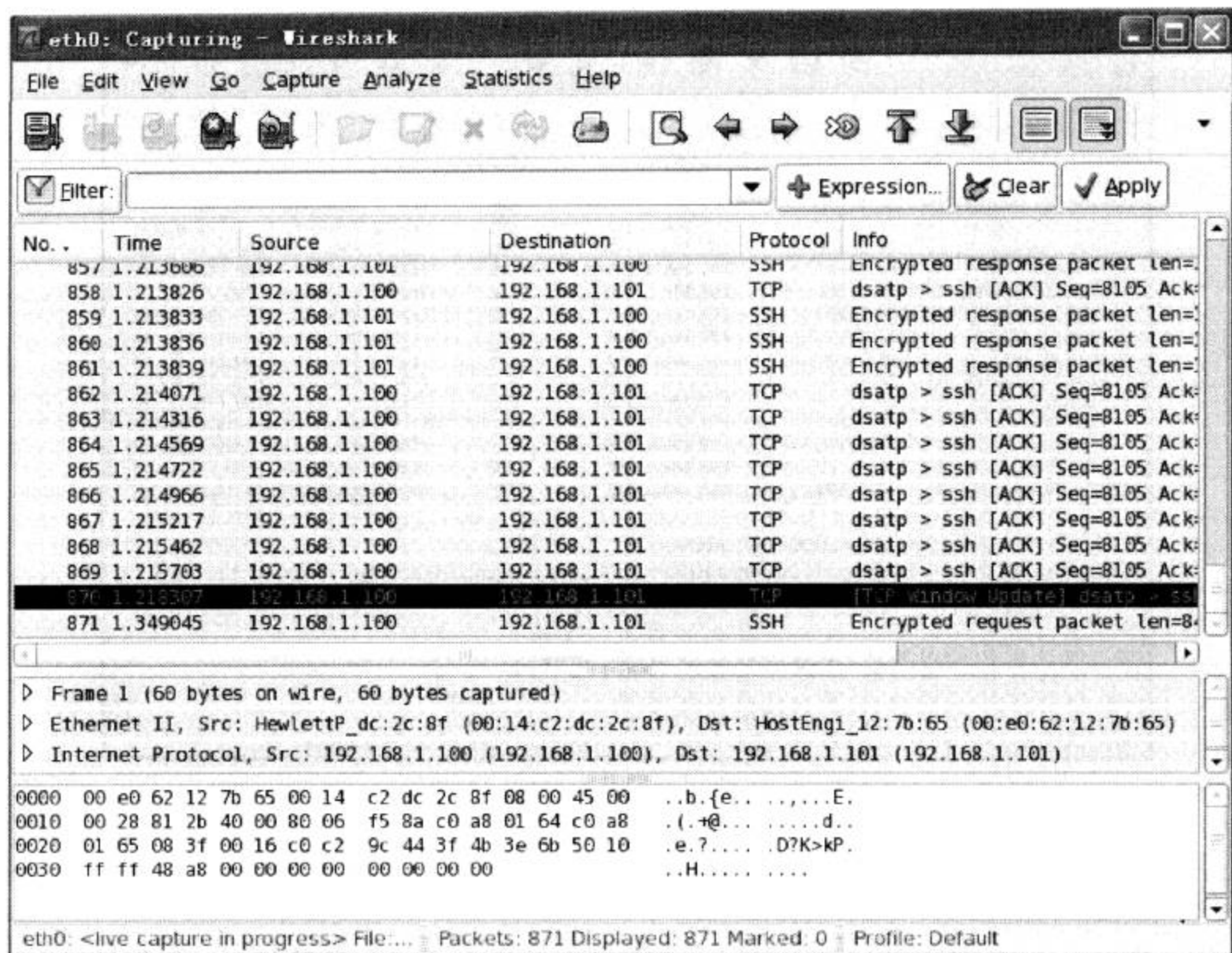


图8-6 Wireshark 抓包结果

抓包结果大家自行分析。在实际工作中大家可以根据当前服务器的实际状态来选择采用何种抓包工具，有图形化界面的我们可以选择 Wireshark，没有的我们选择 TCPDump，我建议将以上两种工具都掌握，因为它们的语法规则基本是类似的。有兴趣的朋友可以直接在规则里输入 icmp，这是运行 ping 的协议，然后在另外的机器上 ping 通 Server，这可以更直观地看到数据的走向，这对于我们以后编写 iptables 脚本大有帮助。这里跟大家交流一下 Wireshark 的一个使用经验，如果我们不知道某项服务要用到哪些协议，哪些端口，可用 Wireshark 抓下包。比如若想知道 Samba 到底占用哪个端口，就可以尝试使用 Wireshark 抓图，然后得出正确的结论。这样是不是很方便呢？大家写抓包规则时，要排除掉 22 的干扰，我提示一下规则，如下：

```
host 192.168.1.100 and (host 192.168.1.101 and port not 22)
```

抓图结果如图8-7所示。通过数据包的分析，我们得出结论：Samba 服务占用的端口为 445，所以只需要在 iptables 防火墙上开放 445 端口即可，命令如下：

```
iptables -A INPUT -p tcp --dport 445 -j ACCEPT
```

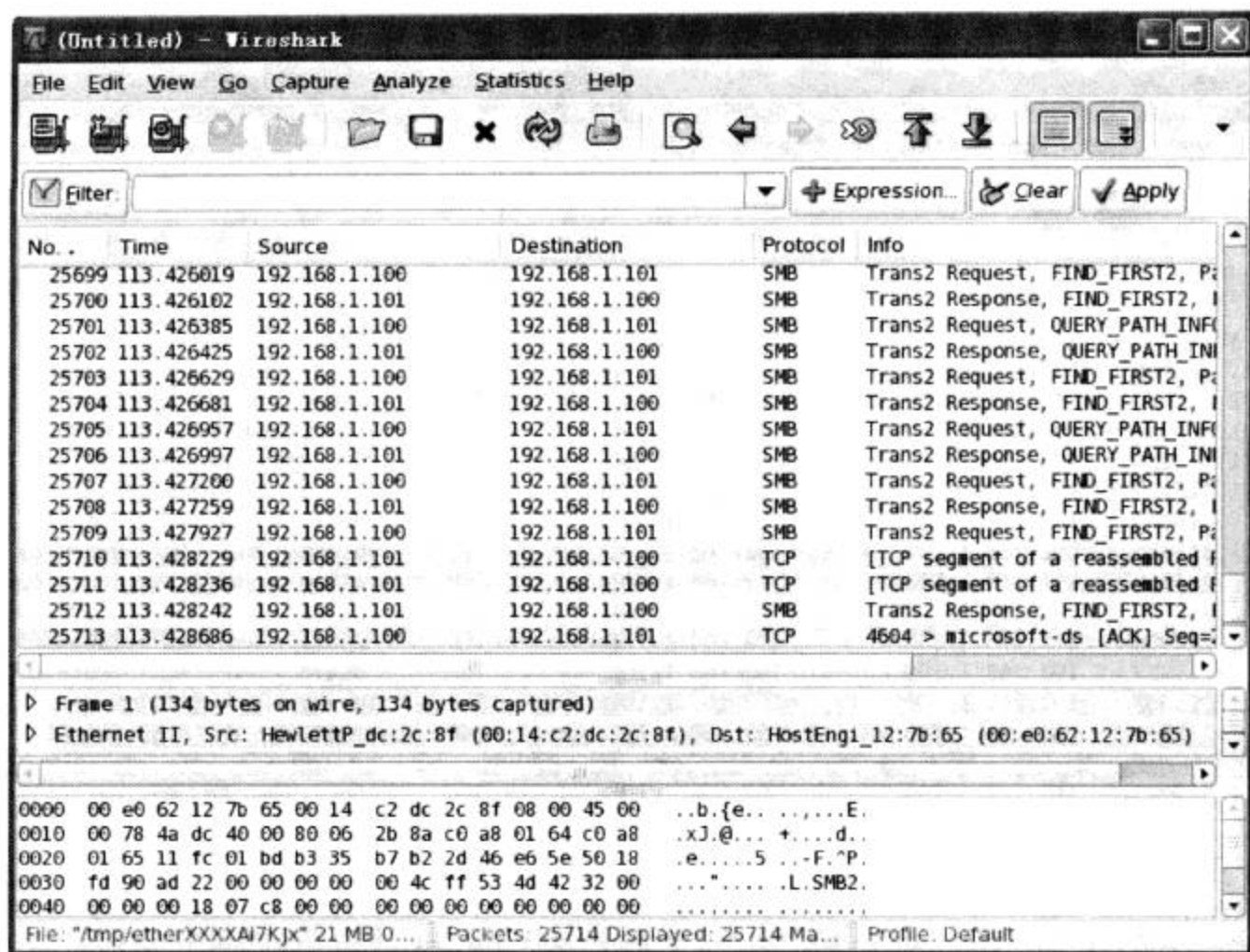


图 8-7 Wireshark 抓 Samba 数据包结果

8.7.3 强大的命令行扫描工具 Nmap

Nmap 用于允许系统管理员查看一个大的网络系统有哪些主机，以及其上运行何种服务。它支持多种协议的扫描，如 UDP、TCP connect()、TCP SYN (half open)、ftp proxy (bounce attack)、Reverse-ident、ICMP (ping sweep)、FIN、ACK sweep、Xmas Tree、SYN sweep 和 Null 扫描。Nmap 还提供了一些实用功能，比如通过 TCP/IP 来甄别操作系统类型、秘密扫描、动态延迟和重发、平行扫描，通过并行的 ping 监测下属的主机、欺骗扫描、端口过滤探测、直接的 RPC 扫描、分布扫描、灵活的目标选择及端口的描述。它的扫描功能异常强大，以至于人们叫它扫描之王。

1. 安装 Nmap

Nmap 要用到一个称为“Windows 包捕获库”的驱动程序 WinPcap——如果你经常从网上下载流媒体电影，可能已经很熟悉这个驱动程序了——某些流媒体电影的地址是加密的，监测这些电影的真实地址就要用到 WinPcap。WinPcap 的作用是帮助调用程序（即这里的 Nmap）捕获通过网卡传输的原始数据。WinPcap 的最新版本在 <http://netgroup-serv.polito.it/winpcap> 上，支持 XP/2K/Me/9x 全系列操作系统，下载得到的是一个执行文件，双击安装，所有选项都确认使用默认设置就可以了，安装好之后需要重新启动。除了命令行版本之外，www.insecure.org 还提供了一个带 GUI 的 Nmap 版本。和其他常见的 Windows 软件一样，GUI 版本需要安装，GUI 版本的功能基本上和命令行版本一样，鉴于许多人更喜欢用命令行版本，下面的说明就以命令行版本为主。在 Centos5.5 下

安装 Nmap 很简单，直接用如下命令即可：

```
yum -y install nmap
```

2. 常用的扫描类型

解开 Nmap 命令行版的压缩包之后，进入 Windows 的命令控制台，再转到安装 Nmap 的目录（如果经常要用 Nmap，最好把它的路径加入到 PATH 环境变量中）。不带任何命令行参数运行 Nmap，Nmap 显示出命令语法，Linux 下是 `nmap --help`（以下命令行操作均适用于 Centos、FreeBSD 和 Windows 系列）。

下面是 Nmap 支持的 4 种最基本的扫描方式：

- ☐ TCP connect() 端口扫描（-sT 参数、-sP 用于扫描整个局域网段）
- ☐ TCP 同步（SYN）端口扫描（-sS 参数）
- ☐ UDP 端口扫描（-sU 参数）
- ☐ TCP ACK 扫描（-sA 参数）

TCP SYN 扫描不太好理解，但如果将它与 TCP connect() 扫描进行比较，就很容易看出这种扫描方式的特点。在 TCP connect() 扫描中，扫描器利用操作系统本身的系统调用打开一个完整的 TCP 连接，也就是说，扫描器打开了两个主机之间的完整握手过程（SYN、SYN-ACK 和 ACK）。一次完整执行的握手过程表明远程主机端口是打开的。TCP SYN 扫描创建的是半打开的连接，它与 TCP connect() 扫描的不同之处在于，TCP SYN 扫描发送的是复位（RST）标记而不是结束 ACK 标记（即 SYN、SYN-ACK 或 RST），如果远程主机正在监听且端口是打开的，远程主机用 SYN-ACK 应答，Nmap 发送一个 RST；如果远程主机的端口是关闭的，它的应答将是 RST，此时 Nmap 转入下一个端口。TCP SYN 的扫描速度要超过 TCP connect() 扫描。如果采用默认计时选项，在 LAN 环境下扫描一个主机，ping 扫描耗时不到 10 秒，TCP SYN 扫描需要大约 13 秒，而 TCP connect() 扫描耗时最多，需要大约 7 分钟。需要说明的是，TCP SYN 扫描又叫隐蔽扫描，扫描时可隐藏自身 IP，因为它很少在目标机上留下记录，三次握手的过程从来都不会完全实现。

3. 命令行参数说明

Nmap 支持丰富、灵活的命令行参数。例如，如果要扫描 192.168.7 网络，可以用 192.168.7.x/24 或 192.168.7.0-255 的形式指定 IP 地址范围。指定端口范围使用 -p 参数，如果不指定要扫描的端口，Nmap 默认扫描从 1 到 1024 再加上 nmap-services 列出的端口，`nmap -sS -O 192.168.0.1` 这样的命令可以对此主机进行操作系统识别。

如果要查看 Nmap 运行的详细过程，只要启用 verbose 模式，即加上 -v 参数，或者加上 -vv 参数可获得更加详细的信息，命令如下：

```
nmap -sS 192.168.7.1-255 -p 20,21,53-110,30000 --v
```

这表示执行一次 TCP SYN 扫描，启用 verbose 模式，要扫描的网络是 192.168.7，检测 20、21、53 到 110 及 30000 以上的端口（指定端口清单时中间不要插入空格）。再举一个例子如下：

```
nmap -sS 192.168.7.1/24 -p 80
```

它扫描 192.168.0 子网，查找在 80 端口监听的服务器（通常是 Web 服务器）。

有些网络设备，例如路由器和网络打印机，可能会禁用或过滤掉某些端口，从而禁止对该设备或跨越该设备的扫描。初步侦测网络情况时，`-host_timeout <毫秒数>` 参数很有用，它表示超时时间，例如 `nmap -sS host_timeout 10000 192.168.0.1` 命令规定的超时时间是 10 000 毫秒。

网络设备上被过滤掉的端口一般会大大延长监测时间，设置超时参数有时可以显著降低扫描网络所需的时间。Nmap 会显示出哪些网络设备响应超时，这时你就可以对这些设备个别处理，从而保证大范围网络扫描的整体速度。当然，`host_timeout` 到底可以节省多少扫描时间，最终还是由网络上被过滤的端口数量决定的。

4. 注意事项

也许你对其他端口扫描器比较熟悉，但 Nmap 绝对值得一试。建议先用 Nmap 扫描一个熟悉的系统，感觉一下 Nmap 的基本运行模式，待熟悉之后，再将扫描范围扩大到其他系统。首先扫描内部网络看看 Nmap 报告的结果，然后从一个外部 IP 地址扫描，注意防火墙、入侵检测系统 (IDS) 及其他工具对扫描操作的反应。通常，`TCP connect()` 会引起 IDS 系统的反应，但 IDS 不一定会记录俗称为“半连接”的 TCP SYN 扫描。最好将 Nmap 扫描网络的报告整理存档，以便随后参考。

使用 Nmap 有几个注意事项如下：

1) 避免误解。不要随意选择测试 Nmap 的扫描目标。许多单位把端口扫描视为恶意行为，所以测试 Nmap 最好在内部网络进行。如有必要，应该告诉同事你正在试验端口扫描，因为扫描可能引发 IDS 警报或其他网络问题。如果不是在你控制的网络、系统及站点上用该工具，你应该首先看许可权。记住，尊重他人网络和系统的隐私意味着别人以后也会这样对你。

2) 关闭不必要的服务。根据 Nmap 提供的报告（同时考虑网络的安全要求），关闭不必要的服务，或者调整路由器的访问控制规则 (ACL)，禁用网络开放给外界的某些端口。

3) 建立安全基准。在 Nmap 的帮助下加固网络，在搞清楚哪些系统和服务可能受到攻击之后，下一步是从这些已知的系统和服务出发建立一个安全基准，以后如果要启用新的服务或服务器，就可以方便地根据这个安全基准来执行。

8.7.4 安全工具 hping

hping 是一个基于命令行的 TCP/IP 工具，它在 UNIX 上得到了很好的应用。不过它并非仅仅是一个 ICMP 请求/响应工具，它还支持 TCP、UDP、ICMP、RAW-IP 协议以及一个路由模型。hping 一直被用作安全工具，可以用来测试网络及主机的安全。它可以发送自定义的 ICMP、UDP 和 TCP 数据包，并接收所有的反馈信息。它的灵感来源于 ping 命令，但其功能远远超过 ping。它还包含一个小型的路由跟踪模块，并支持 IP 分段。此工具可以在常用工具无法对有防火墙保护的主机进行路由跟踪并进行 ping 探测时大显身手。它经常可以帮助您找出防火墙的规则集，当然还可以通过它来学习 TCP/IP 协议，并做一些 IP 协议的实验。其官方网址为：<http://www.hping.org>。

hping 在 Centos 及 FreeBSD 下的安装

我手上正好有两台 Centos5.0 i386 的机器，我就拿出其中一台安装 hping（大家这里注意下 32 位系统与 64 位系统的区别）。

首先我们必须安装一些基础库，命令如下：


```
yum -y install gcc libpcap-devel
```

然后下载源码包，再进行编译安装，如下所示：

```
cd /usr/local/src
wget http://www.hping.org/hping3-20051105.tar.gz
tar zxvf hping3-20051105.tar.gz
cd hping3-20051105
./configure && make && make install
```

进行到 make 这步时会出现如下报错：

```
iftop.c:22:21: error: net/bpf.h: No such file or directory
make [2]: *** [iftop.o] Error 1
make [2]: Leaving directory '/usr/local/src/iftop-1.0pre1'
make [1]: *** [all-recursive] Error 1
make [1]: Leaving directory '/usr/local/src/iftop-1.0pre1'
make: *** [all] Error 2
```

系统找不到 bpf.h 库文件，解决方法如下：

```
ln -s /usr/include/pcap.h /usr/include/net/bpf.h
```

做完这步后，hping 就可以安装成功了。它现在的最新版本是 hping3，我们执行 hping 和 hping2 时其实所执行的都是 hping3 命令，大家可以注意观察一下，命令如下：

```
ls -lsart hping*
```

此命令显示结果如下：

```
360 -rwxr-xr-x 1 root root 356963 Jun 1 17:31 hping3
  4 lrwxrwxrwx 1 root root    16 Jun 1 17:31 hping2 -> /usr/sbin/hping3
  4 lrwxrwxrwx 1 root root    16 Jun 1 17:31 hping -> /usr/sbin/hping3
```

但在 Centos5.5 x86_64 的机器上安装 hping 时始终报错，出错信息如下：

```
In file included from ars.h:20,
      from arsglue.c:7:
bytesex.h:22:3: error: #error can not find the byte order for this architecture, fix bytesex.h
In file included from arsglue.c:7:
ars.h:190:2: error: #error "Please, edit Makefile and add -DBYTE_ORDER_(BIG|LITTLE)_ENDIAN"
ars.h:254:2: error: #error "Please, edit Makefile and add -DBYTE_ORDER_(BIG|LITTLE)_ENDIAN"
ars.h:323:2: error: #error "Please, edit Makefile and add -DBYTE_ORDER_(BIG|LITTLE)_ENDIAN"
make: *** [arsglue.o] Error 1
```

而我更换了一台 FreeBSD8.1x86_64 的机器，在其下用 pkg_add 安装没有任何问题，命令如下：

```
pkg_add -r -v hping
```

此命令显示结果如下：

```
extract: Package name is hping-2.0.0r3,1
extract: CWD to /usr/local
extract: /usr/local/sbin/hping
```

```

extract: /usr/local/man/man8/hping.8.gz
extract: /usr/local/share/doc/hping/AS - BACKDOOR
extract: /usr/local/share/doc/hping/HPING2 - HOWTO.txt
extract: /usr/local/share/doc/hping/HPING2 - IS - OPEN
extract: /usr/local/share/doc/hping/MORE - FUN - WITH - IPID
extract: /usr/local/share/doc/hping/SPOOFED_SCAN.txt
extract: /usr/local/share/doc/hping/APD.txt
extract: CWD to .
Runningmtree for hping-2.0.0r3,1..
mtree -U -f +MTREE_DIRS -d -e -p /usr/local >/dev/null
Attempting to record package into /var/db/pkg/hping-2.0.0r3,1..
Package hping-2.0.0r3,1 registered in/var/db/pkg/hping-2.0.0r3,1

```

说明, hping 工具很顺利就在这台 FreeBSD8.1x86_64 安装成功了。我们可以用此工具来制作 Smurf 攻击, 用以下命令:

```
hping2 -1 -a 192.168.1.101 192.168.1.255
```

Smurf 攻击属于 DoS 攻击, 该攻击是攻击者以一个伪造的源地址向一个或多个广播地址发送 ICMP Echo 请求, 这个伪造地址所属的主机正是被攻击的对象, 该主机将被来自其他网络的广播地址主机的 Echo 应答信息所淹没。想象一下一台拥有很窄且很慢 Internet 连接的机器接收到来自 254 台主机的 Echo 应答信息的情景, 再想象一下它所接收的是来自 100 个网络且每个网络都拥有 254 台主机的 Echo 应答信息的情况, 可以断定, 这样不断接收 ICMP 应答信息, 不需多时整个网络将瘫痪。

现在还没有一个很好的基于主机的防御方法来抵御 Smurf 攻击, 就算是独立主机将接收 ICMP 应答信息的功能禁用, 网络带宽仍将被大量进入网络的应答信息所消耗。一个更好的解决方法是尽可能地作为回应的 Echo 应答进行限速, 同时允许为了诊断问题而进行的正当应答。推荐以下做法:

```
iptables -A INPUT -p icmp --icmp-type echo-request -m limit --limit 1/s --limit-burst 10
-j ACCEPT
```

这句话限制每秒的 ping 请求及回应最多不能超过每秒 10 个, 这可以有效地防止 Smurf 这类 ping 的洪水攻击, 也允许了正常的 ping 请求。

ping 洪水攻击不等于 Ping of Death, 后者是一种较老的攻击方式, 它是通过发送很大的 ping 数据包来实现的。有弱点的系统将在这种攻击中崩溃, Linux 及许多当前的 Unix 操作系统都没有此弱点。

8.8 iptables 的简单脚本学习

这一节我们通过编写几个简单的 iptables 脚本来熟悉 iptables 语法规则。网络拓扑很简单, iptables 本身机器的 IP 为: 192.168.1.101/24, 另一台机器的 IP 为: 192.168.1.102。

注意 为了大家学习方便, 我加载了 ip_conntrack 模块, 方便追踪数据包的流向, 但是在生产环境下不建议大家开启此模块。

8.8.1 普通的 Web 主机防护脚本

普通的 Web 主机防护脚本比较容易实现，Web 主机主要开放两个端口：80 和 22，其他端口则关闭。另外由于这里没有涉及多少功能，所以模块的载入也很简单，只涉及 filter 表，而且脚本的初始化也很简单。

我们可以按照编写 iptables 的流程顺序来写脚本，脚本内容如下：

```
#/bin/bash
iptables -F
iptables -X
iptables -Z

modprobe ip_tables
modprobe iptable_nat
modprobe ip_nat_ftp
modprobe ip_conntrack

iptables -P INPUT DROP
iptables -P FORWARD ACCEPT
iptables -P OUTPUT ACCEPT

iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

iptables -A INPUT -p tcp -m multiport --dports 22,80 -j ACCEPT
iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
```

iptables 脚本开启后，我们可以用命令查看一下结果，如下所示：

```
iptables -nv -L
```

此命令显示结果如下：

```
Chain INPUT (policy DROP 13539 packets, 763K bytes)
  pkts bytes target    prot opt in     out     source            destination
    0    0 ACCEPT    all  --  lo      *        0.0.0.0/0         0.0.0.0/0
  480 32744 ACCEPT    tcp  --  *        *        0.0.0.0/0         0.0.0.0/0      multiport
dports 22,80
   13  1411 ACCEPT    all  --  *        *        0.0.0.0/0         0.0.0.0/0      state RE-
LATED,ESTABLISHED

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target    prot opt in     out     source            destination

Chain OUTPUT (policy ACCEPT 472 packets, 52779 bytes)
  pkts bytes target    prot opt in     out     source            destination
    0    0 ACCEPT    all  --  *        lo      0.0.0.0/0         0.0.0.0/0
```

而原来正常提供 Samba 文件服务的，在开启此脚本后就不能提供服务了，看来脚本生效了。Windows 下的报错界面如图 8-8 所示。



图 8-8 原本正常的 Samba 服务给出了报错信息

我们在另一台机器上开启 Nmap 扫描，发现 Samba 端口被 iptables 屏蔽了，命令如下：

```
nmap -sT 192.168.1.101
```

此命令显示结果如下：

```
Starting Nmap 4.11 (http://www.insecure.org/nmap/) at 2011-05-25 00:46 CST
Interesting ports on 192.168.1.101:
Not shown: 1678 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
MAC Address:00:E0:62:12:7B:65 (Host Engineering)
Nmap finished: 1 IP address (1 host up) scanned in 37.721 seconds
```

8.8.2 如何让别人 ping 不到自己而自己能 ping 通别人

我们如何通过 iptables 来控制让别人不能 ping 通我们的主机，而我们的主机却能 ping 通别人呢？达到此目的所牵涉的表和链不多，脚本也进行了简化处理，代码如下：

```
#!/bin/bash
iptables -F
iptables -F -t nat
iptables -X
modprobe ip_tables
modprobe iptable_nat
modprobe ip_nat_ftp
modprobe ip_nat_irc
modprobe ip_conntrack
modprobe ip_conntrack_ftp

iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
```



```

iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -p tcp -m multiport --dport 80,22 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 0 -j ACCEPT

iptables -A OUTPUT -o lo -j ACCEPT
iptables -A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -p tcp -m multiport --sport 80,22 -j ACCEPT
iptables -A OUTPUT -p icmp --icmp-type 8 -j ACCEPT

```

执行此脚本后，我们还是照常用 iptables-nv -L 查看执行后的结果，命令如下：

```
iptables -nv -L
```

此命令显示如果如下：

```

Chain INPUT (policy DROP 71 packets, 5964 bytes)
pkts bytes target      prot opt in     out    source            destination
    0     0 ACCEPT      all  --  lo     *       0.0.0.0/0         0.0.0.0/0
  312 21904 ACCEPT      all  --  *      *       0.0.0.0/0         0.0.0.0/0      state RE-
LATED,ESTABLISHED
    0     0 ACCEPT      tcp  --  *      *       0.0.0.0/0         0.0.0.0/0      multiport
dports 80,22
    0     0 ACCEPT      icmp --  *      *       0.0.0.0/0         0.0.0.0/0      icmp type 0
Chain FORWARD (policy DROP 0 packets, 0 bytes)
pkts bytes target      prot opt in     out    source            destination
Chain OUTPUT (policy DROP 4 packets, 288 bytes)
pkts bytes target      prot opt in     out    source            destination
    0     0 ACCEPT      all  --  *      lo     0.0.0.0/0         0.0.0.0/0
  253 28748 ACCEPT      all  --  *      *       0.0.0.0/0         0.0.0.0/0      state RE-
LATED,ESTABLISHED
    0     0 ACCEPT      tcp  --  *      *       0.0.0.0/0         0.0.0.0/0      multiport
sports 80,22
    1    84 ACCEPT      icmp --  *      *       0.0.0.0/0         0.0.0.0/0      icmp type 8

```

开启另一台服务器 192.168.1.102，然后互相 ping 一下试试，我们会发现，192.168.1.101 是可以 ping 通 192.168.1.102 的，但 192.168.1.102 上面却 ping 不通 192.168.1.101，命令如下：

```
ping 192.168.1.101
```

此命令执行结果如下：

```

PING 192.168.1.101 (192.168.1.101) 56 (84) bytes of data.
ping: sendmsg: Network is unreachable
ping: sendmsg: Network is unreachable
ping: sendmsg: Network is unreachable
ping: sendmsg: Network is unreachable

```

我们可以在 192.168.1.101 上抓包试一下，用 Wireshark 或 TCPDump 均可。TCPDump 命令

如下：

```
tcpdump host 192.168.1.101 and 192.168.1.102 -vv
```

结果如下：

```
16:34:59.337811 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto: ICMP (1), length: 84) 192.168.1.102 > 192.168.1.101: ICMP echo request, id 50211, seq 472, length 64
16:35:01.010688 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto: ICMP (1), length: 84) 192.168.1.102 > 192.168.1.101: ICMP echo request, id 50211, seq 473, length 64
16:35:02.670812 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto: ICMP (1), length: 84) 192.168.1.102 > 192.168.1.101: ICMP echo request, id 50211, seq 474, length 64
16:35:04.324072 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto: ICMP (1), length: 84) 192.168.1.102 > 192.168.1.101: ICMP echo request, id 50211, seq 475, length 64
16:35:06.120890 arp who-has 192.168.1.101 tell 192.168.1.102
16:35:06.120908 arp reply 192.168.1.101 is-at 00:e0:62:12:7b:65 (oui Unknown)
```

在 ICMP 协议里，icmp-type 为 8 表示 ping request，即 ping 请求，而 icmp-type 为 0 表示 echo relay，即回显应答。ICMP 也是 TCP/IP 协议的一种，它也是需要三次握手的。192.168.1.102 向 192.168.1.101 发送 ping request 请求，但收不到 192.168.1.101 的 echo relay 消息，三次握手完成不了，所以 192.168.1.102 的机器 ping 不通 192.168.1.101，这样就达到了我们要实现的目的了。

8.8.3 建立安全的 vsftpd 服务器

vsftpd 有两种运行模式，一种是主动模式（ACTIVE），一种是被动模式（即大家非常熟悉的 PASV）。它们的工作机制又是怎样的呢？

21 端口是 FTP 的命令传输端口，而数据的传输分为主动模式和被动模式。主动模式下的工作原理为：

客户端向服务器的 FTP 端口（默认是 21）发送连接请求，服务器接受连接，建立一条命令链路。当需要传送数据时，客户端在命令链路上用 PORT 命令告诉服务器：“我打开了 xx 端口，你过来连接我。”于是服务器从 20 端口向客户端的 xx 端口发送了连接请求，建立了一条数据链路来传送数据。

在 Centos5.5 下 vsftpd 默认是以被动模式启动的，相对于主动模式而言，被动模式要复杂些。

这种被动方式是为了解决服务器发起到客户连接的问题而开发的一种不同的 FTP 连接方式，或者叫做 PASV，当客户端通知服务器它处于被动模式时才会启用。

在被动方式 FTP 中，命令连接和数据连接都由客户端发起，这样就可以解决从服务器到客户端数据端口的入方向连接被防火墙过滤掉的问题。它的工作原理是：当开启一个 FTP 连接时，客户端打开两个任意的非特权本地端口（ $N > 1024$ 和 $N + 1$ ）。第一个端口连接服务器的 21 端口，但与主动方式的 FTP 不同的是，客户端不会提交 PORT 命令并允许服务器来回连接它的数据端口，而是提交 PASV 命令。这样做的结果是服务器会开启一个任意的非特权端口（端口大于 1024），并发送 PORT 命令给客户端。然后客户端发起从本地端口 $N + 1$ 到服务器端口的连接，以传送数据。

对于服务器端的防火墙来说，必须允许下面的通信才能支持被动方式的 FTP：

□ 可从任何端口到服务器的 21 端口（客户端初始化的连接 Client→Server）。

- 可从服务器的 21 端口到任何大于 1024 的端口（服务器响应到客户端的控制端口的连接 Server→Client）。
- 可从任何端口到服务器的大于 1024 的端口（客户端初始化数据连接到服务器指定的任意端口 Client→Server）。
- 可从服务器的大于 1024 的端口到远程的大于 1024 的端口（服务器发送 ACK 响应和数据到客户端的数据端口 Server→Client）。

请大家注意选择模式的原则，如下所示：

- Client 没有防火墙时，用主动模式连接即可。
- Server 没有防火墙时，用被动模式即可。
- 双方都有防火墙时，vsftpd 设置被动模式高端口范围，Server 打开那段范围，Client 用被动模式连接即可。

是否采取被动模式取决于客户程序，在 Windows XP 命令行模式下使用 FTP 命令连接 FTP 服务器，用的是主动模式；在浏览器方式下连接 FTP 服务器时，可以修改访问所使用的模式，如图 8-9 所示。

我们在 Windows XP 的命令行模式用 FTP 命令登录 vsftpd 服务器，如图 8-10 所示。

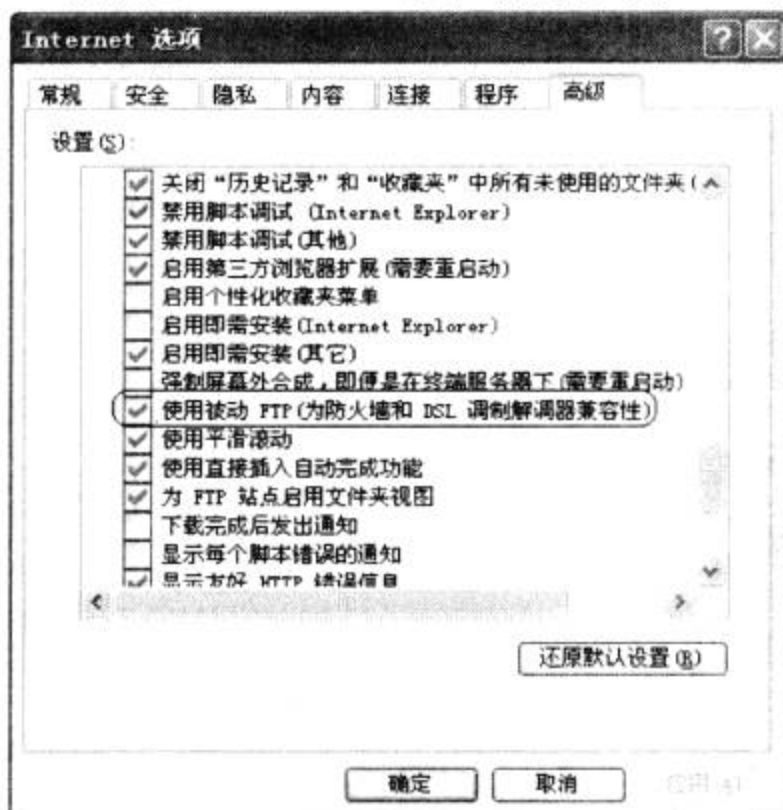


图 8-9 Windows XP 下的 IE 可设置 FTP 的被动模式

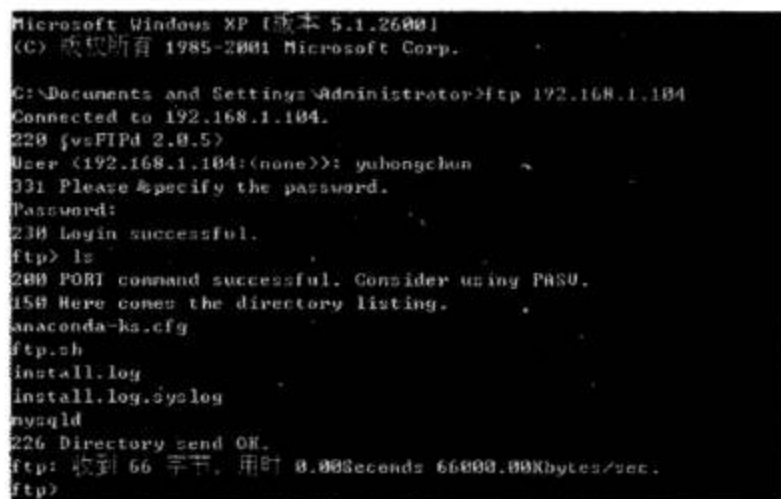


图 8-10 Windows XP 用命令行登录 vsftpd 服务器

注意 我们通过 Wireshark 抓包分析时，发现服务器的 21 和 20 端口都有数据传输，vsftpd 服务器用的主动传输模式。很多网上教程都说 LITERAL PASV 或 passive 可以改变命令行下的 FTP 主/被模式，但我通过抓包分析发现这根本就行不通，尤其是 passive 命令，执行都执行不了。但通过 Windows XP 的 IE 来进行 FTP 的数据传输，确实使用的是被动模式。有兴趣的朋友也可以研究一下在命令行下如何通过命令切换主/被动模式。

我们再来看一下与其相关的脚本，代码如下：

```
#!/bin/bash
iptables -F
```

```
iptables -X
iptables -Z
```

```
modprobe ip_conntrack_ftp
modprobe ip_nat_ftp
```

```
#FTP 需要 ip_nat_ftp 模块
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD ACCEPT
```

#这里为了安全,OUTPUT 我也定义为 DROP

```
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
```

#打开回环口,免得不必要的麻烦

```
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 22 -j ACCEPT
iptables -A INPUT -p tcp --dport 21 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 21 -j ACCEPT
```

#22 和 21 端口都打开,确认这两个端口的数据都会被顺利放行

```
iptables -A INPUT -p tcp -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -p tcp -m state --state ESTABLISHED,RELATED -j ACCEPT
```

#与 FTP-DATA 有关的 RELATED 包都会被放行,用状态来约束数据包比用端口智能些

执行脚本后,我们照例查看执行后的 iptables 结果,命令如下:

```
iptables -nv -L
```

此命令显示结果如下:

```
Chain INPUT (policy DROP 2 packets, 305 bytes)
pkts bytes target    prot opt in     out    source          destination
  0     0 ACCEPT    all  --  lo     *       0.0.0.0/0       0.0.0.0/0
 70   5160 ACCEPT    tcp  --  *      *       0.0.0.0/0       0.0.0.0/0      tcp dpt:22
 13    639 ACCEPT    tcp  --  *      *       0.0.0.0/0       0.0.0.0/0      tcp dpt:21
  6    336 ACCEPT    tcp  --  *      *       0.0.0.0/0       0.0.0.0/0      state RELAT-
ED,ESTABLISHED

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target    prot opt in     out    source          destination

Chain OUTPUT (policy DROP 2 packets, 144 bytes)
pkts bytes target    prot opt in     out    source          destination
  0     0 ACCEPT    all  --  *      lo     0.0.0.0/0       0.0.0.0/0
 70   7580 ACCEPT    tcp  --  *      *       0.0.0.0/0       0.0.0.0/0      tcp spt:22
 13    837 ACCEPT    tcp  --  *      *       0.0.0.0/0       0.0.0.0/0      tcp spt:21
 10   1104 ACCEPT    tcp  --  *      *       0.0.0.0/0       0.0.0.0/0      state RELAT-
```


ED, ESTABLISHED

我们可以用 X-manager3.0 的 Xftp (它默认也是采用被动模式) 来尝试 FTP 传输数据, 如图 8-11 所示。当然大家也可以用自己熟悉的 Cute-FTP 或 FileZilla (开源软件) 来进行 FTP 数据传输, 效果如图 8-12 所示。上传和下载均没什么问题, 这证明脚本确实有效果 (此脚本限制比较严格, INPUT 和 OUTPUT 默认均是 DROP 的)。

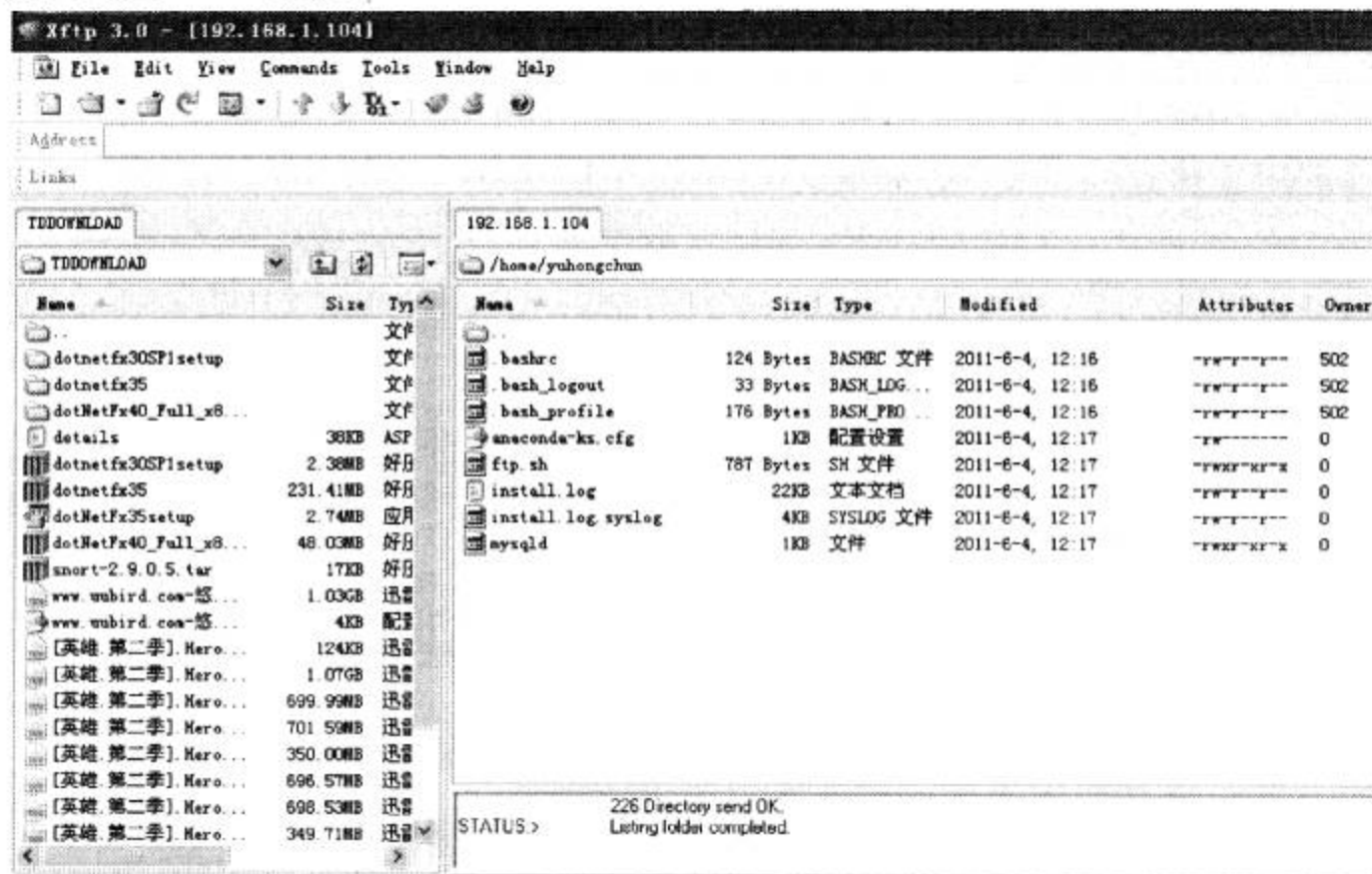


图 8-11 X-manager3.0 中的 Xftp 运行界面

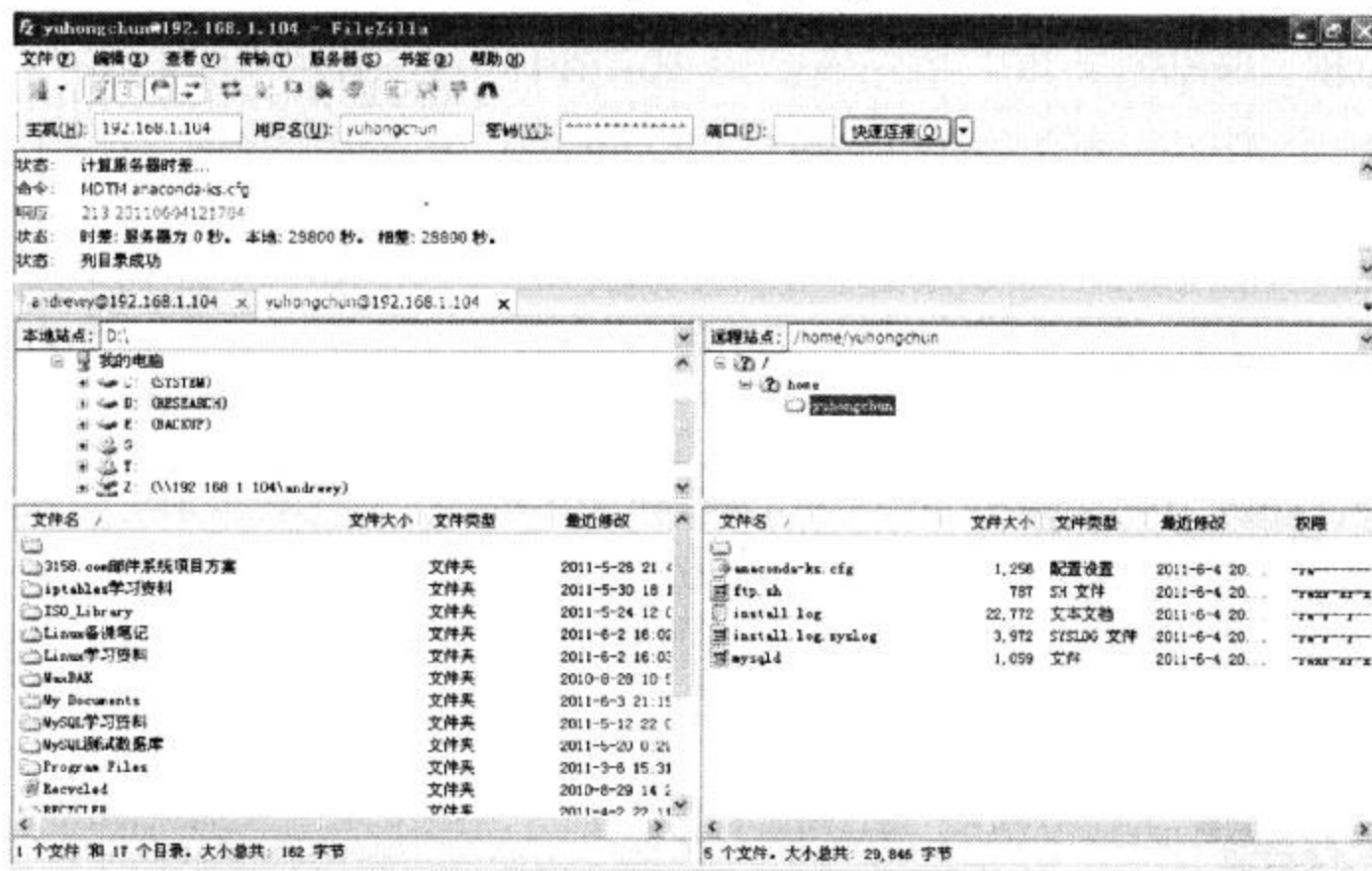


图 8-12 FileZilla FTP Client 运行界面

下面用 Wireshark 抓一下 vsftpd 的数据传输信息，如图 8-13 所示。

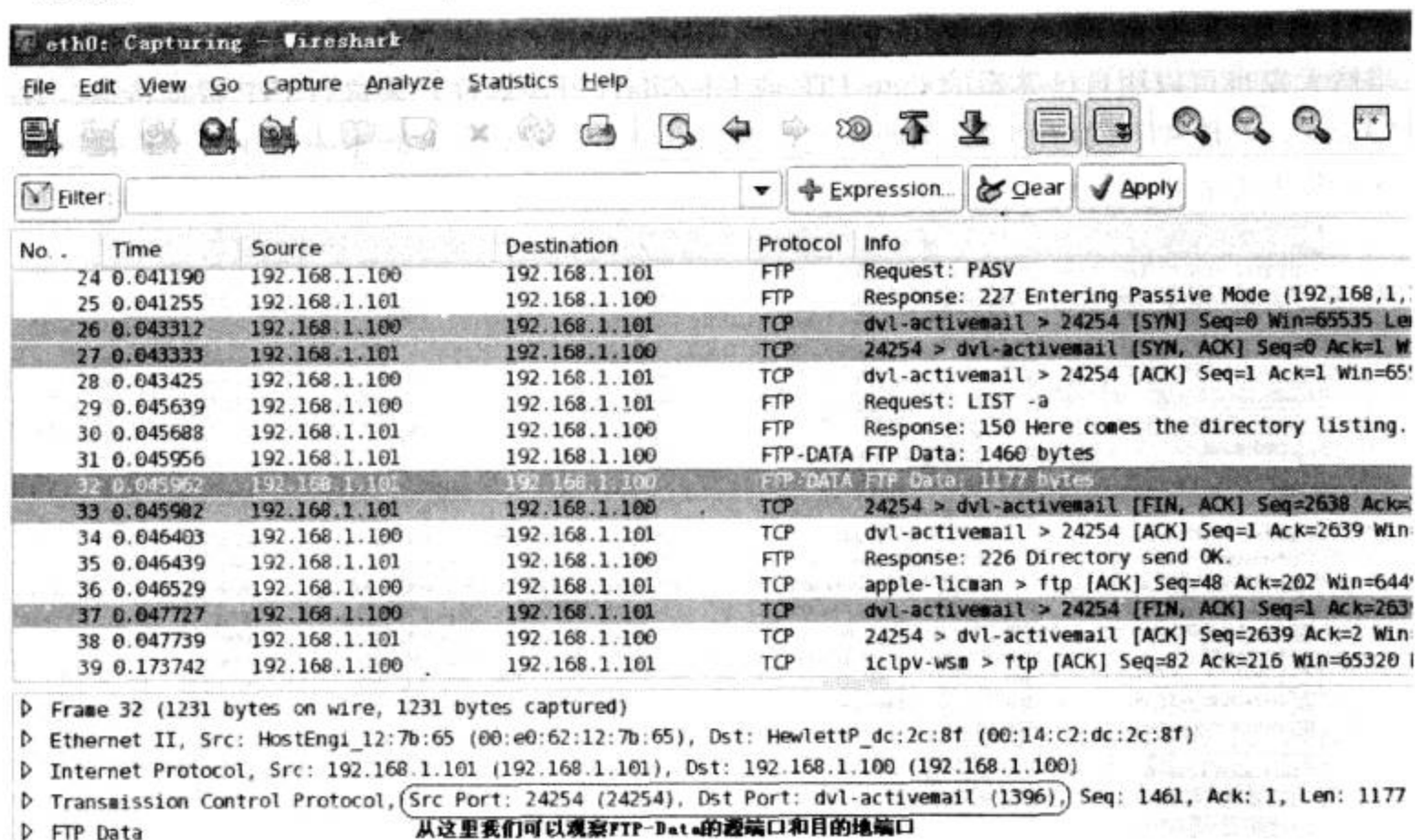


图 8-13 用 Wireshark 抓 vsftpd 数据包

大家注意看一下，FTP-Data 数据包的源端口和目的地端口都是大于 1024 的，由于还是多端口传输数据，如果这个地方采用端口号的方式来做会很麻烦，例如以下语句：

```
iptables -A INPUT -p tcp --sport 1024: --dport 1024: -j ACCEPT
iptables -A OUTPUT -p tcp --sport 1024: --dport 1024: -j ACCEPT
```

如果加上这两句话，相当于开放了 1024 以上的所有端口，那还有何安全可言？如果在这种情况下要对端口进行限制，就需要在 vsftpd 配置多个端口，然后再分别针对这些端口进行允许的操作。但是如果用状态来设置的话一切都会显得简单方便多了，所以这也是大家公认 iptables 的 state 防火墙更智能的原因。这个脚本比较安全，大家可以用之来架设内部 FTP。如果要将此 FTP 置于公网上，注意硬件防火墙与之搭配的问题（尤其是 vsftpd 的主/被动问题）。

8.9 线上生产服务器的 iptables 脚本

在编写安全的 iptables 脚本之前，我们依旧要先做好准备工作。现在比较新的 Centos 系统版本是 Centos5.6 x86_64（期待 Centos6.0 x86_64），我们可以通过如下命令查看它自带的 iptables 版本：

```
iptables --version
iptables v1.3.5
```

如果要查看系统的内核版本号，可以用如下命令：

```
uname -r
2.6.18-238.el5
```

为什么这里要采用较新的 Centos 系统呢,这是因为 iptables 现在有许多新的模块,如果在原有的老系统和老内核上采用这些新的 iptables 模块,则必须要采取重新编译内核的方法。不过现在新版的 Centos 系统自带的 iptables v1.3.5 已经支持了原先不支持的许多模块,比如 connlimit 和 recent 模块,所以我们部署 iptables 防火墙时就容易多了。随着 RHEL6.1 的成功发行,Centos6.0 的时代也将会来临,等不及的朋友也可以先尝试一下 RHEL6.1x86_64。

注意 鉴于 iptables v1.3.5 和 iptables v1.4.x 的语法有细微的区别,以下的所有脚本都以 Centos5.6 x86_64 系统、内核为 2.6.18-238.el5、iptables v1.3.5 为平台来说明。

在调试 iptables 脚本之前,由于这里的服务器都涉及生产服务器,为了防止发生意外事件,所以配置了一个 crontab 计划任务,每 5 分钟关闭一次防火墙,免得将自己千里之外的防火墙 SSH 连接都断掉,那样就得不偿失了。crontab 计划任务如下:

```
* /5 * * * * root /etc/init.d/iptables stop
```

等确保 iptables 万无一失以后,我们再清除掉此计划任务。

8.9.1 安全的主机 iptables 防火墙脚本

下面以我自己的 iRedmail 邮件服务器举例说明之,系统的默认策略是 INPUT 为 DROP, OUTPUT、FORWARD 链为 ACCEPT。DROP 设置得比较宽松,因为我们知道出去的数据包是安全的。

脚本代码如下所示:

```
#!/bin/bash
iptables -F
iptables -F -t nat
iptables -X

iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT

#load connection-tracking modules

modprobe iptable_nat
modprobe ip_conntrack_ftp
modprobe ip_nat_ftp

iptables -A INPUT -f -m limit --limit 100/sec --limit-burst 100 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type echo-request -m limit --limit 1/s --limit-burst 10 -j ACCEPT
iptables -A INPUT -p tcp -m tcp --tcp-flags SYN,RST,ACK SYN -m limit --limit 20/sec --limit-burst 200 -j ACCEPT

iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -p tcp -m multiport --dport 80,443,25,465,110,995,143,993,587,465,22 -
```

j ACCEPT

成功运行此脚本后系统应该是不会报错的，命令如下：

```
iptables -nv -L
```

此命令显示结果如下：

```
Chain INPUT (policy DROP 0 packets, 0 bytes)
  pkts bytes target    prot opt in     out     source            destination
    0    0 ACCEPT    all  -f  *      *        0.0.0.0/0         0.0.0.0/0         limit: avg
100/sec burst 100
    0    0 ACCEPT    tcp  --  *      *        0.0.0.0/0         0.0.0.0/0         tcp flags:
0x16/0x02 limit: avg 20/sec burst 200
    0    0 ACCEPT    all  --  lo     *        0.0.0.0/0         0.0.0.0/0
    26 1925 ACCEPT    all  --  *      *        0.0.0.0/0         0.0.0.0/0         state RE-
LATED, ESTABLISHED
    0    0 ACCEPT    tcp  --  *      *        0.0.0.0/0         0.0.0.0/0         multiport
dports 80,443,25,465,110,995,143,993,587,465,22

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
  pkts bytes target    prot opt in     out     source            destination
    0    0 ACCEPT    icmp --  *      *        0.0.0.0/0         0.0.0.0/0         icmp type 8
limit: avg 1/sec burst 10
Chain OUTPUT (policy ACCEPT 20 packets, 1873 bytes)
  pkts bytes target    prot opt in     out     source            destination
    0    0 ACCEPT    all  --  *      lo        0.0.0.0/0         0.0.0.0/0
```

在主机的防护上我们配置了一些安全措施，以防止外部的 ping 和 SYN 洪水攻击，并且考虑到外部的疯狂端口扫描软件可能会影响服务器的入口带宽，所以在这里也做了限制。命令如下所示：

```
iptables -A INPUT -p tcp --syn -m limit --limit 100/s --limit-burst 100 -j ACCEPT
```

上面的命令每秒钟最多允许 100 个新连接，请注意这里的新连接指的是 state 为 New 的数据包。在后面我们也配置了允许状态为 ESTABLISHED 和 RELATED 的数据通过。另外，100 这个阈值则要根据服务器的实际情况来调整，如果是并发量不大的服务器这个数值就要调小；如果是访问量非常大且并发数不小的服务器，这个值则还需要调大。再看以下命令：

```
iptables -A INPUT -p icmp --icmp-type echo-request -m limit --limit 1/s -limit-burst 10 -j ACCEPT
```

这是为了防止 ping 洪水攻击，限制每秒的 ping 包不超过 10 个。

```
iptables -A INPUT -p tcp -m tcp --tcp-flags SYN,RST,ACK SYN -m limit --limit 20/sec --limit-burst 200 -j ACCEPT
```

上面的命令防止各种端口扫描，将 SYN 及 ACK SYN 限制为每秒钟不超过 200 个，免得把服务器带宽耗尽了。

iptables 防火墙运行后，我们可以运行 Nmap 工具进行扫描，命令如下：

```
nmap -P0 -sS 211.143.6.X
```

此命令的执行结果如下：


```

Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2009-03-29 16:21 CST
Interesting ports on 211.143.6.X:
Not shown: 1668 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
80/tcp    open  http
110/tcp   open  pop3
111/tcp   open  rpcbind
143/tcp   open  imap
443/tcp   open  https
465/tcp   open  smtps
587/tcp   open  submission
993/tcp   open  imaps
995/tcp   open  pop3s
1014/tcp  open  unknown

```

在这里，我们发现一个 1014 端口被某个进程打开了，用 `lsof -i: 1014` 查看发现又是 `rpc.statd` 打开的，这个服务每次用的端口都不一样啊！本来想置之不理的，但是如果 `rpc.statd` 不能正确处理 `SIGPID` 信号，远程攻击者可利用这个漏洞关闭进程，进行拒绝服务攻击，所以还是得想办法解决它。我们发现 `rpc.statd` 是由服务 `nfslock` 开启的，进一步查询得知它是一个可选的进程，它允许 NFS 客户端在服务器上对文件加锁。这个进程对应于 `nfslock` 服务，于是我们关掉了此服务，命令如下所示：

```

service nfslock stop
chkconfig nfslock off

```

8.9.2 自动分析黑名单及白名单的 iptables 脚本

本 `iptables` 脚本是一个自动分析黑名单和白名单的安全脚本，脚本路径为 `/root/deny_100.sh`。运行此脚本时我们要注意以下几点：

1) 此脚本能自动过滤掉企业中通过 NAT 出去的白名单 IP，很多中小企业都是通过 `iptables` 作为 NAT 软路由上网，我们可以将一些与我们有往来的公司及本公司的安全 IP 添加进白名单，以防误剔。

2) 这里定义的阈值 `DEFIIN` 是 100，其实这个值应该根据具体生产环境而定，50 ~100 之间较好。

3) 此脚本原理其实很简单，通过判断瞬间连接数是否大于 100 来抉择，如果是白名单里的 IP 则跳过；如果不是，则用 `iptables -I` 参数将此恶意 IP 禁掉。这里建议不要用 `-A`，`-A` 在 `iptables` 的规则里是最后添加的，往往达不到即时剔除的效果。大家都知道，`iptables` 中针对链的操作其实是有规则编号的，`-I` 是在规则的最前面插入的，而 `iptables` 是按照规则的顺序来生效的，所以可以采用 `-I` 实现立即禁止某 IP 的目的。

4) 此脚本是在线上的邮件服务器上进行调试的，最后更新时间为 2010 年 5 月 24 日。

脚本代码如下，我们可以用 `cat` 命令来查看脚本内容：

```
cat /root/deny_100.sh
```

`/root/deny_ 100.sh` 脚本的内容如下：

```

#!/bin/bash
netstat -an | grep :25 | grep -v 127.0.0.1 | awk '{ print $5 }' | sort | awk -F: '{print $1, $4}' |
uniq -c | awk '$1 > 100 {print $1, $2}' > /root/black.txt

for i in `awk '{print $2}' /root/black.txt`
do
COUNT=`grep $i /root/black.txt | awk '{print $1}'`
DEFINE="100"
ZERO="0"
if[ $COUNT -gt $DEFINE];
then
grep $i /root/white.txt > /dev/null
if[ $? -gt $ZERO];
then
echo "$COUNT $i"
iptables -I INPUT -p tcp -s $i -j DROP
fi
fi
done

```

2009 年 3 月 30 日 14:25, 用下列命令监控:

```
netstat -an | grep :25 | grep -v 127.0.0.1 | awk '{print $5}' | sort | awk -F: '{print $1}' | uniq -
c | awk '$1 > 100'
```

此命令执行后显示的内容如下:

```
1122 219.136.163.207
17 61.144.157.236
```

219.136.163.207 这个 IP 的瞬间连接数为 1122, 这个值明显不正常, 这极有可能是一个攻击 IP, 我们用 <http://www.ip138.com> 一查, 发现了如下结果:

ip138.com IP 查询 (搜索 IP 地址的地理位置)

您查询的 IP: 219.136.163.207

本站主数据: 广东省广州市 电信 (荔湾区)

参考数据一: 广东省广州市 电信 (荔湾区)

参考数据二: 广东省广州市荔湾区 电信 ADSL

调用 deny_100.sh 将此 IP 禁止, 再运行 ./root/count.sh 后则无显示了, 表明脚本执行成功, 可用 iptables-nL 验证。另外我们要允许此脚本每 10 分钟执行一次, 命令如下:

```
*/10 * * * * root /bin/sh /root/deny_100.sh
```

一般来说, 10 分钟或更长时间执行一次此脚本是没有问题的, 因为此脚本只要发现有可疑 IP 大量连接的情况, 在排除是白名单的情况下会立即禁止此 IP 访问。

注意 有的朋友喜欢用 while 循环的方法, 在这里也可以用, 但要记得注意防止出现死循环的问题。

在内网下模拟测试攻击方法如下:

我们可以在本机 iptables 上防火墙 (Server 机器 IP 为 192.168.1.101) 上, 将此脚本监听端口

由 25 改成 80 端口，执行命令 `service httpd start` 开启 Web 服务，记得要执行 `crontab` 计划任务。然后在另一台 Centos5.5 机器上，比如 192.168.1.102 上安装 `webbench` 软件，用它来模拟 80 端口的攻击，先进入 `/usr/local/src` 目录下，然后下载并安装 `webbench`，如下所示：

```
wget http://blog.s135.com/soft/linux/webbench/webbench-1.5.tar.gz
```

然后解压缩此软件，然后编译安装，如下所示：

```
tar zxvf webbench-1.5.tar.gz
cd webbench
make && make install
```

然后我们可以用以下命令进行压力测试（模拟端口攻击），命令如下：

```
webbench -t 1000 -c 150 http://192.168.1.101
```

此行的意思是在 1000 秒的时间内，模拟 150 个并发去访问 `http://192.168.1.101` 的网站。

我们在服务器上可以执行以下命令，观测 `iptables` 执行结果：

```
iptables -nv -L
Chain INPUT (policy ACCEPT 849K packets, 57M bytes)
pkts bytes target      prot opt in      out     source      destination
1161 85180 DROP        tcp  --  *      *        192.168.1.102  0.0.0.0/0
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in      out     source      destination
Chain OUTPUT (policy ACCEPT 849K packets, 92M bytes)
pkts bytes target      prot opt in      out     source      destination
```

这时在 192.168.1.102 的机器上，我用 `Elinks` 访问不了 192.168.1.101 的 Web 服务了，证明此脚本成功运行了。有兴趣的朋友也可以依照以上步骤进行实验。

此脚本在线上实际使用的结果如下：

我们在将此脚本放在线上服务器上使用时，发现对于非 Web 的应用服务器，比如 Mail、DNS 确实有效果，而对于 Web 应用服务器效果则不是特别明显。这是因为现在许多 Web 服务器，特别是存储海量图片小文件的服务器，如果我们单击一个链接，这个链接可能同时会产生许多对应页面的链接，而这个 IP 在 `netstat` 中对应的链接数就有 100 多个，而脚本则认为此 IP 就应该是一个危险的 IP，应该 DROP 掉它。但我们通过 `Nginx` 的日志分析发现，这个客户端的 IP 是一个正常的客户 IP，所以将此脚本应用于 Web 服务器不太合适。我目前也只将其用于邮件服务器和 DNS 服务器，请大家在实际工作中也注意甄别使用，如果确实需要限制 IP 在单位时间内的连接次数，可以利用 `iptables` 的 `recent` 模块来进行，下一节将会跟大家详细说明。

8.9.3 利用 recent 模块限制同一 IP 的连接数

上一节向大家演示了自动区别黑名单和白名单的 `iptables` 脚本，发现将其应用于 Web 服务器的效果并不是太好。如果想限制瞬间连接数过大的恶意 IP 地址，可以考虑使用 `iptables` 的模块 `recent`。

新版的 `iptables` 有个好用又有效率的功能，可以用它阻止瞬间联机太多的来源 IP。这种阻挡功能在某些地方很受欢迎，比如说某大型讨论区网站，每个网页都遭到无聊人士的连接，一瞬间太多的链接访问导致服务器呈现呆滞状态。

这时，就需要使用下列 3 行指令，代码如下：


```
iptables -I INPUT -p tcp --dport 80 -m state --state NEW -m recent --name web --set
iptables -A INPUT -m recent --update --name web --seconds 60 --hitcount 20 -j LOG --log -
prefix 'HTTP attack: '
iptables -A INPUT -m recent --update --name web --seconds 60 --hitcount 20 -j DROP
```

第一行：-I 表示将本规则插入到 INPUT 链的最上头。这是什么样的规则呢？iptables 判断只要是 TCP 性质的联机，目标端口是 80 并且目标 IP 是我们机器 IP 的联机，就将这个联机列入这份 Web 清单中。

第二行：-A 表示将本规则附在 INPUT 链的最尾端。如果在 60 秒内，同一个来源连续产生了多个联机，在到达第 20 个联机时，对此联机留下 Log 记录。记录行会以 HTTP attack 开头。

第三行：-A 表示将本规则附在 INPUT 链的最尾端。在与第二行同样的条件下，本次的动作则是将此联机给断掉，即将每 60 秒内有 20 个联机的数据包给 DROP 掉。

所以，这三行规则表示，我们允许一个客户端每分钟内可以接上服务器的 20 个新连接，具体数值可以由具体的生产环境决定。这些规则也可以用在其他对 Internet 开放的应用服务上，例如 Mail 邮件服务器和 DNS 解析服务器。

为什么新版的 iptables 在阻挡上很有效率呢？因为在旧版的 iptables 中，并没有这些新模块功能，导致我们需要使用操作系统的 SHELL（比如 netstat）接口，周期性地执行网络检查与拦阻动作。前者只动用到网络层的资源，而后者已经是应用层的大量（相对而言）运算。试想，服务器资源都已经被非法客户端消耗殆尽了，哪还有余力周期性地呼叫软件层级的计算，来阻挡非法客户端呢？新版的 iptables 增加了此模块后就可以直接禁止恶意 IP，而不需要调用操作系统的 SHELL 接口，所以更有效率。

接下来我们可以测试一下这个脚本的效果，步骤如下。

1) 将这三句话写成脚本形式，方便执行操作，代码如下：

```
iptables -I INPUT -p tcp --dport 80 -m state --state NEW -m recent --name web --set
iptables -A INPUT -m recent --update --name web --seconds 60 --hitcount 20 -j LOG --log -
prefix 'HTTP attack: '
iptables -A INPUT -m recent --update --name web --seconds 60 --hitcount 20 -j DROP
```

执行后我们照例查看结果，命令如下：

```
iptables -nv -L
Chain INPUT (policy ACCEPT 4556 packets, 259K bytes)
pkts bytes target      prot opt in      out     source        destination
9180 551K      tcp -- *      *       0.0.0.0/0     0.0.0.0/0     tcp dpt:80
state NEW recent: SET name: web side: source
10003 636K LOG      all -- *      *       0.0.0.0/0     0.0.0.0/0     recent:
UPDATE seconds: 60 hit_count: 20 name: web side: source LOG flags 0 level 4 prefix 'HTTP attack: '
10003 636K DROP    all -- *      *       0.0.0.0/0     0.0.0.0/0     recent:
UPDATE seconds: 60 hit_count: 20 name: web side: source
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target      prot opt in      out     source        destination
Chain OUTPUT (policy ACCEPT 6829 packets, 6947K bytes)
pkts bytes target      prot opt in      out     source        destination
```

2) 在另一台 192.168.1.102 的机器上面我们可以运行 webbench 压力测试工具，来模拟非法攻

击，命令如下：

```
webbench -t 1000 -c 500 http://192.168.1.101/
Webbench - Simple Web Benchmark 1.5
Copyright (c) Radim Kolar 1997 - 2004, GPL Open Source Software.
Benchmarking: GET http://192.168.1.101/
500 clients, running 1000 sec.
```

3) 观察结果可以得知，在相当长的时间内（即时间为 1000 秒的时间段内），IP 地址为 192.168.1.102 的机器是访问不了 192.168.1.101 的 Web 服务的，只有过了这段压力测试的时间后，192.168.1.102 的机器才能正常访问 192.168.1.101 的 Web 服务，证明脚本生效。我们还可以查看服务器的 iptables 日志，输入以下命令即可查看系统的最后 100 条日志（iptables 日志默认是放在 /var/log/messages 里）：

```
tail -n100 /var/log/messages
```

结果如下，我们监测到大量带有 HTTP attack 日志头的 iptables 日志，如果发现此 IP 数量重复数量非常之多，则直接用 iptables-I 命令将其禁止掉，省得它下次又重新发包攻击。

```
May 31 12:06:24 iptables kernel: HTTP attack: IN=eth0 OUT= MAC=00:e0:62:12:7b:65:00:0c:29:7e:
82:2d:08:00 SRC=192.168.1.102 DST=192.168.1.101 LEN=117 TOS=0x00 PREC=0x00 TTL=64 ID=58051 DF
PROTO=TCP SPT=1585 DPT=80 WINDOW=730 RES=0x00 ACK PSH URGP=0
May 31 12:06:24 iptables kernel: HTTP attack: IN=eth0 OUT= MAC=00:e0:62:12:7b:65:00:0c:29:7e:
82:2d:08:00 SRC=192.168.1.102 DST=192.168.1.101 LEN=117 TOS=0x00 PREC=0x00 TTL=64 ID=65165 DF
PROTO=TCP SPT=1586 DPT=80 WINDOW=730 RES=0x00 ACK PSH URGP=0
May 31 12:06:24 iptables kernel: HTTP attack: IN=eth0 OUT= MAC=00:e0:62:12:7b:65:00:0c:29:7e:
82:2d:08:00 SRC=192.168.1.102 DST=192.168.1.101 LEN=117 TOS=0x00 PREC=0x00 TTL=64 ID=27276 DF
PROTO=TCP SPT=1587 DPT=80 WINDOW=730 RES=0x00 ACK PSH URGP=0
May 31 12:06:24 iptables kernel: HTTP attack: IN=eth0 OUT= MAC=00:e0:62:12:7b:65:00:0c:29:7e:
82:2d:08:00 SRC=192.168.1.102 DST=192.168.1.101 LEN=117 TOS=0x00 PREC=0x00 TTL=64 ID=37974 DF
PROTO=TCP SPT=1588 DPT=80 WINDOW=730 RES=0x00 ACK PSH URGP=0
May 31 12:06:24 iptables kernel: HTTP attack: IN=eth0 OUT= MAC=00:e0:62:12:7b:65:00:0c:29:7e:
82:2d:08:00 SRC=192.168.1.102 DST=192.168.1.101 LEN=117 TOS=0x00 PREC=0x00 TTL=64 ID=61898 DF
PROTO=TCP SPT=1590 DPT=80 WINDOW=730 RES=0x00 ACK PSH URGP=0
May 31 12:06:38 iptables kernel: HTTP attack: IN=eth0 OUT= MAC=00:e0:62:12:7b:65:00:0c:29:7e:
82:2d:08:00 SRC=192.168.1.102 DST=192.168.1.101 LEN=117 TOS=0x00 PREC=0x00 TTL=64 ID=21600 DF
PROTO=TCP SPT=2554 DPT=80 WINDOW=730 RES=0x00 ACK PSH URGP=0
May 31 12:06:38 iptables kernel: HTTP attack: IN=eth0 OUT= MAC=00:e0:62:12:7b:65:00:0c:29:7e:
82:2d:08:00 SRC=192.168.1.102 DST=192.168.1.101 LEN=117 TOS=0x00 PREC=0x00 TTL=64 ID=3782 DF
PROTO=TCP SPT=2555 DPT=80 WINDOW=730 RES=0x00 ACK PSH URGP=0
```

这里值得注意的是，iptables 的 recent 模块功能虽然强大，但它并不适用于有些基于 LVS + Keepalived 的 Linux 集群环境，这是因为后端的 Web 都是通过前端的 LVS 负载均衡器来连接的，有时遇到并发数比较大的情况，比如单位时间内的并发数超过 2000，这时候用 recent 模块肯定是不行的。另外为了数据包转包的高效，我们往往要在 LVS + Keepalived 架构中关掉 iptables 防火墙。另外大家都知道，LVS/DR 是基于公网地址的，现在 SSH 暴力破解工具比比皆是，我们又应该如何防止 SSH 暴力破解呢？

8.9.4 利用 DenyHosts 工具和脚本来防止 SSH 暴力破解

我在对 Nagios 外网监控服务器进行测试时设置的密码是 redhat_123456，可放进公网的第一天就被人把密码改了，郁闷！后来环境部署成熟以后发现仍然有不少外网 IP 在扫描和试探，看来不用点工具不行啊！于是我想到了 DenyHosts，它是用 Python 2.3 写的一个程序，它会分析/var/log/secure 等日志文件，当发现同一 IP 在进行多次 SSH 密码尝试时就会将 IP 记录到/etc/hosts.deny 文件，从而达到自动屏蔽该 IP 的目的。

DenyHosts 官方网址：<http://denyhosts.sourceforge.net>

安装 DenyHosts 的详细步骤如下。

1. 检查安装条件

1) 判断系统安装的 sshd 是否支持 tcp_wrappers（默认都支持），命令如下：

```
ldd /usr/sbin/sshd | grep libwrap.so.0
libwrap.so.0 => /lib64/libwrap.so.0 (0x00002ba0aecb6000)
```

2) 判断默认安装的 Python 版本，命令如下：

```
python -V
Python 2.4.3
```

Centos5.5 x86 | Centos5.6 x86_64 均已默认安装了 Python 2.4.3。

2. 安装及配置 DenyHosts 工具

在确认系统已安装 Python 2.3 以上版本的情况下，执行以下步骤：

1) 安装 DenyHosts。

```
# cd /usr/local/src
# wget http://jaist.dl.sourceforge.net/sourceforge/denyhosts/DenyHosts-2.6.tar.gz
# tar xzf DenyHosts-2.6.tar.gz
# cd DenyHosts-2.6
# python setup.py install
```

程序脚本自动安装到/usr/share/denyhosts。

库文件自动安装到/usr/lib/python2.3/site-packages/DenyHosts。

denyhosts.py 自动安装到/usr/bin。

2) 设置启动脚本，命令如下所示：

```
# cd /usr/share/denyhosts/
# cp daemon-control-dist daemon-control
# chown root daemon-control
# chmod 700 daemon-control
# grep -v "^#" denyhosts.cfg-dist > denyhosts.cfg
# vim denyhosts.cfg
```

我们可以根据自己的需要对文件 denyhosts.cfg 进行相应的修改，如下所示：

```
SECURE_LOG = /var/log/secure
```

表示 RedHat/Centos 系统中安全日志的文件位置。

其他版本的 Linux 请大家根据 denyhosts.cfg-dist 内的提示进行选择。

```
PURGE_DENY = 30m
```

表示过多久后清除。

```
DENY_THRESHOLD_INVALID = 1
```

表示允许无效用户 (/etc/passwd 未列出) 登录失败的次数。

```
DENY_THRESHOLD_VALID = 5
```

表示允许有效 (普通) 用户登录失败的次数。

```
DENY_THRESHOLD_ROOT = 3
```

表示允许 root 登录失败的次数。

```
HOSTNAME_LOOKUP = NO
```

表示是否做域名反解。

3) 启动 DenyHosts 工具, 命令如下:

```
/usr/share/denyhosts/daemon-control start
```

如果要使 DenyHosts 每次重启后自动启动还需做如下设置:

```
# cd /etc/init.d
ln -s /usr/share/denyhosts/daemon-control denyhosts
chkconfig --add denyhosts
chkconfig --level 345 denyhosts on
```

然后就可以启动了, 启动命令如下:

```
service denyhosts start
```

DenyHosts 配置文件的语法如下:

```
SECURE_LOG = /var/log/secure
```

此乃 SSH 日志文件, 它是根据这个文件来判断非法 IP 的。

```
HOSTS_DENY = /etc/hosts.deny
```

这是控制用户登录的文件。

```
PURGE_DENY = 5m
```

表示要过多久后清除已经禁止的 IP, 这个可根据具体时间而定。

```
BLOCK_SERVICE = sshd
```

表示禁止的服务名。

```
DENY_THRESHOLD_INVALID = 1
```

表示允许无效用户失败的次数。

```
DENY_THRESHOLD_VALID = 10
```

表示允许普通用户登录失败的次数。

```
DENY_THRESHOLD_ROOT = 5
```

表示允许 root 登录失败的次数。

```
HOSTNAME_LOOKUP = NO
```

表示是否做域名反解。

```
DAEMON_LOG = /var/log/denyhosts
```

这是自己的日志文件。

```
ADMIN_EMAIL = yuhongchun027@163.com
```

这是管理员邮件地址，它会给管理员发邮件，Centos5.5 默认是开启了 sendmail 邮件服务的。

下面是全自动下载安装的小脚本 install_denyhosts.sh（以下脚本在 Centos5.1 | 5.2 | 5.5 | 5.6 下测试通过）。当然安装后还得手动调整配置文件。脚本代码如下所示：

```
#!/bin/bash
cd /usr/local/src
wget
http://jaist.dl.sourceforge.net/sourceforge/denyhosts/DenyHosts-2.6.tar.gz
tar zxf DenyHosts-2.6.tar.gz
cd DenyHosts-2.6
python setup.py install
cd /usr/share/denyhosts/
cp daemon-control-dist daemon-control
chown root daemon-control
chmod 700 daemon-control
grep -v "^#" denyhosts.cfg-dist > denyhosts.cfg
echo "/usr/share/denyhosts/daemon-control start" >> /etc/rc.local
cd /etc/init.d
ln -s /usr/share/denyhosts/daemon-control denyhosts
chkconfig --add denyhosts
chkconfig --level 345 denyhosts on
service denyhosts start
```

下面是 DenyHosts 的示例，如果在/etc/hosts.deny 里有记录的 IP 机器仍然想连接安装了 DenyHosts 的机器的话，则会被拒绝，如下所示：

```
[root@ autolemp ~]# ssh 192.168.0.154
root@ 192.168.0.154's password:
Permission denied, please try again.
root@ 192.168.0.154's password:
Permission denied, please try again.
root@ 192.168.0.154's password:
Permission denied (publickey,gssapi-with-mic,password)
```

出现上面最后这行代码表示/etc/hosts.deny 文件生效了，DenyHosts 工具部署成功。

DenyHosts 的原理很简单，它其实就是收集/var/log/secure 的信息，如果 root 登录失败的次数超过 10 次，它就会将其写进/etc/hosts.deny 文件里。事实上，我们完全可以用 SHELL 来做这些事情，

比如说不想对生产服务器进行太多改动的时候，像我的线上服务器有许多是基于LVS环境的，我不想安装DenyHosts工具来防止SSH暴力破解，所以最后采用了SHELL。下面是脚本/root/ssh_deny.sh的内容（此脚本已在Centos5.5 x86_64的机器上通过）：

```
#!/bin/bash
cat /var/log/secure |awk '/Failed/{print $(NF-3)}' |sort |uniq -c |awk '{print $2"="$1;}'
>/root/black.txt
DEFINE="100"
for i in `cat /root/black.txt`
do
    IP=`echo $i |awk -F= '{print $1}'`
    NUM=`echo $i |awk -F= '{print $2}'`
    if [ $NUM -gt $DEFINE ];
    then
        grep $IP /etc/hosts.deny > /dev/null
        if [ $? -gt 0 ];
        then
            echo "sshd:$IP" >> /etc/hosts.deny
        fi
    fi
done
```

将此脚本写进Crontab计划任务里，每一分钟执行一次，我们在/etc/crontab文件的最后一行添加如下内容：

```
* /1 * * * root sh /root/ssh_deny.sh
```

初期我写此脚本时想用时间来收集信息，即每隔一段时间就收集一次/var/log/secure的日志信息。后来觉得这个方法比较繁琐，所以我就采取了上面所用的方法。这里稍微解释一下上面的脚本，它其实是放在Crontab里的，每隔一分钟就会去读取一下/var/log/secure的日志信息，定义连接root的IP阈值为100，如果某IP在/etc/hosts.deny里，就什么也不做；如果某IP阈值超过10但不在/etc/hosts.deny里（\$?值不为0，即非状态下），就将其添加进去。此脚本放在线上环境执行一段时间后我发现进行SSH暴力破解的恶意IP明显减少了，以下是公网IP地址为203.93.236.146的Web机器执行此脚本一段时间后的/root/black黑名单清单：

```
125.208.5.78=15
204.231.108.140=33
210.66.168.73=1
220.181.121.213=4
221.149.97.116=23
64.64.28.139=12
81.92.157.210=3
82.94.235.23=1
```

罗列部分/etc/hosts.deny里禁止的恶意IP地址，如下所示：

```
sshd:113.130.71.75
sshd:202.29.15.46
sshd:116.236.131.148
sshd:218.38.58.157
```

```
sshd:114.113.148.226
sshd:221.13.29.86
sshd:211.68.70.103
sshd:60.250.109.85
sshd:61.183.16.205
sshd:60.191.239.231
sshd:218.38.16.101
sshd:113.31.18.18
sshd:122.193.5.68
sshd:211.154.153.94
sshd:61.130.156.146
sshd:210.212.190.49
sshd:121.32.133.226
sshd:112.2.0.194
sshd:123.108.110.158
sshd:204.231.108.140
sshd:221.149.97.116
```

使用此脚本的起因是我的线上机器比较多，由于有别的同事需要通过 22 端口进行工作，所以我没有做 22 端口重定向的工作（这也是一项保护 Linux 服务器的措施），这导致网站刚上线时 SSH 端口连接繁多，而我的网站是基于 LVS + Keepalived 的，公司明文规定线上的机器不能做程序级别的改动，但我想防止 SSH 暴力破解，所以我在我的每一台线上机器都部署了此脚本，从而达到防止 SSH 暴力破解的目的。

8.9.5 将 iptables 作为企业的 NAT 路由器

1. 什么是 NAT?

在传统的标准的 TCP/IP 通信过程中，所有的路由器仅仅充当一个中间人的角色，也就是通常所说的存储转发，路由器并不会对转发的数据包进行修改，更为确切地说，除了将源 MAC 地址换成自己的 MAC 地址以外，路由器不会对转发的数据包做任何修改。NAT（Network Address Translation，网络地址翻译）恰恰是出于某种特殊需要而对数据包的源 IP 地址、目的 IP 地址、源端口、目的端口进行改写的操作。请大家注意的是，NAT 不是转发，请大家注意这二者的区别。

2. 为什么要进行 NAT?

我们来看看在什么情况下需要做 NAT。

假设有一家 ISP 提供园区 Internet 接入服务，为了方便管理，该 ISP 分配给园区用户的 IP 地址都是伪 IP，但是部分用户要求建立自己的 WWW 服务器对外发布信息，这时候我们就可以通过 NAT 来提供这种服务了。我们可以在防火墙的外部网卡上绑定多个合法 IP 地址，然后通过 NAT 技术使发给其中某一个 IP 地址的包转发至内部某一用户的 WWW 服务器上，然后再将该内部的 WWW 服务器响应包伪装成合法 IP 发出的包。

再比如使用拨号上网的网吧，因为只有一个合法的 IP 地址，必须采用某种手段让其他机器也可以上网，通常采用的是代理服务器的方式。但是代理服务器，尤其是应用层代理服务器，只能支持有限的协议，如果过了一段时间后又有新的服务出来，则只能等待代理服务器版本升级以支持该新应用。如果采用 NAT 来解决这个问题，因为是在应用层以下进行处理，NAT 不但可以获得很高

的访问速度，而且可以无缝地支持任何新的服务或应用。

还有一个方面的应用就是重定向，也就是在接收到一个包后，不是转发这个包，而是将其重定向到系统上的某一个应用程序。最常见的应用就是和 squid 配合使用成为透明代理，在对 HTTP 流量进行缓存的同时，可以提供对 Internet 的无缝访问。

参考资料：http://tech.ccidnet.com/art/737/20060705/596567_1.html

3. NAT 的类型有哪些？

NAT 分成了两种类型，即源 NAT (SNAT) 和目的 NAT (DNAT)。顾名思义，所谓 SNAT 就是改变转发数据包的源地址，所谓 DNAT 就是改变转发数据包的目的地址。iptables 完全支持这两种 NAT，我们可以在企业内部或网吧通过架设 iptables 服务器充当 NAT 路由器，带动内部所有的机器 NAT 上网。

iptables 作为 NAT 路由器的脚本内容如下（由于此处的 iptables 是置于企业内网，不用担心非法扫描和无聊的数据包，并且此主机的内存比较充裕，这里开启了 ip_conntrack 模块）：

```
#!/bin/bash
echo "1" > /proc/sys/net/ipv4/ip_forward
arp -f /root/mac.txt

modprobe iptable_nat
modprobe ip_conntrack
modprobe ip_conntrack_ftp
modprobe ip_nat_ftp

iptables -F INPUT
iptables -F FORWARD
iptables -F POSTROUTING -t nat

iptables -P FORWARD DROP

cat /root/mac.txt | while read LINE
do
    ipad='echo $LINE | awk '{print $1}''
    macd='echo $LINE | awk '{print $2}''
    iptables -A FORWARD -s $ipad -m mac --mac-source $macd -j ACCEPT
done

iptables -A FORWARD -i eth0 -m state --state ESTABLISHED,RELATED -j ACCEPT
#iptables -t nat -A POSTROUTING -o eth0 -s 192.168.1.102 -j SNAT --to 113.57.224.3
iptables -t nat -A POSTROUTING -o eth0 -s 192.168.1.0/24 -j MASQUERADE
```

此脚本实现功能如下：

- 因为是绑定 MAC 地址上网，企业内部的客户机如果绑定 MAC 可以防范企业内部的 ARP 病毒。
- 对局域网内的机器上网进行严格控制，每增加一台工作用机，就必须重新刷新 NAT 服务器的 ip-mac 对应关系，严格杜绝了公司外来用机上网的问题（有的员工周末喜欢带自己手提电脑来公司上网），在安全问题上做到了防患于未然。
- 配合 NAT 网关服务器的监控软件 NTOP + iptraf，可以做到及时监控每台主机的流量情况，如发现流量异常可及时通知网管或行政处理。
- 经实际使用发现，以此脚本作 NAT 网关路由器时，可将公司 10MB 的电信光纤带宽发挥到

极致，即如果一个员工不限速使用迅雷，整个公司办公环境内将均打不开网页。

- `ipad` 这些变量与“=”号之间不要有空格符，要不然脚本运行时很有可能失败，大家在调试此脚本时也要注意这点。

下面解释一下脚本的重点内容，如下所示：

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

表示开启 IP 路由转发，用于在多块网卡之间流通数据。如果我们的企业内部不止一个网段，则可以在 NAT 路由器上通过增加物理网卡的方式来增加网段。开启 IP 路由转发后，数据包可以在不同网卡之间流通。再看下面的命令：

```
arp -f /root/mac.txt
```

以 `mac.txt` 文件定义的主机 IP 及 MAC 地址来代替原有 `arp` 的对应关系。每增加一台工作用机，就要重新运行一次此脚本，虽然麻烦但对于整个公司的网络安全来说却是至关重要的，后期此工作可以交给网管来做。

```
modprobe iptable_nat
modprobe ip_conntrack
modprobe ip_conntrack_ftp
modprobe ip_nat_ftp
```

表示加载一些需要的模块，可根据需要加载。

```
iptables -F INPUT
iptables -F FORWARD
iptables -F POSTROUTING -t nat
```

清除本网关的 Filter、FORWARD、POSTROUTING 链的默认规则。

```
iptables -P FORWARD DROP
```

将 FORWARD 的默认策略设置为禁止一切（基于最安全原则考虑）。

```
cat /root/mac.txt | while read LINE
do
ipad='echo $LINE | awk '{print $1}''
macd='echo $LINE | awk '{print $2}''
iptables -A FORWARD -s $ipad -m mac --mac-source $macd -j ACCEPT
done
```

客户机须绑定 MAC 地址才能上网，这样可防止恶意增加 IP 在公司内部上网，引起安全隐患。

```
iptables -A FORWARD -i eth0 -m state --state ESTABLISHED,RELATED -j ACCEPT
```

表示网关上有几块网卡，`eth0` 接的是外网 IP 地址，`eth1`、`eth2`、`eth3` 等对应该局域网相应的 IP，用于规划局域中对应的网段。

```
#iptables -t nat -A POSTROUTING -o eth0 -s 192.168.1.0/24 -j SNAT --to 113.57.224.3
iptables -t nat -A POSTROUTING -o eth0 -s 192.168.1.0/24 -j MASQUERADE
```

如果大家是在 ADSL 环境中上网的话，就要用到 MASQUERADE。如果是固定的光纤上网情形，我们将对应的网段 SNAT 给我们的公网 IP 即可，这可以根据具体环境而定。

注意 SNAT 的特殊情况是 IP 欺骗，也就是所谓的 MASQUERADE，通常建议在拨号上网的时候使用，或者说在合法 IP 地址不固定的情况下使用。

8.9.6 如何使用工具精确地监控 NAT 路由器

以 iptables 作 NAT 路由器后，我们需要将局域网内的网关都设置为此机器的 IP，DNS 服务器则可以配置为我们内网的 DNS 或公网 DNS 服务器。NAT 路由器正常工作后，我们需要安装 IPtraf 和 NTOP 对局域网内的流量进行监控，如果发现有可疑 IP 用迅雷等 P2P 软件大量消耗网络带宽，就可以禁止这些 IP 或进行处理。

1. 用 IP 数据包流量实时监控 IPtraf

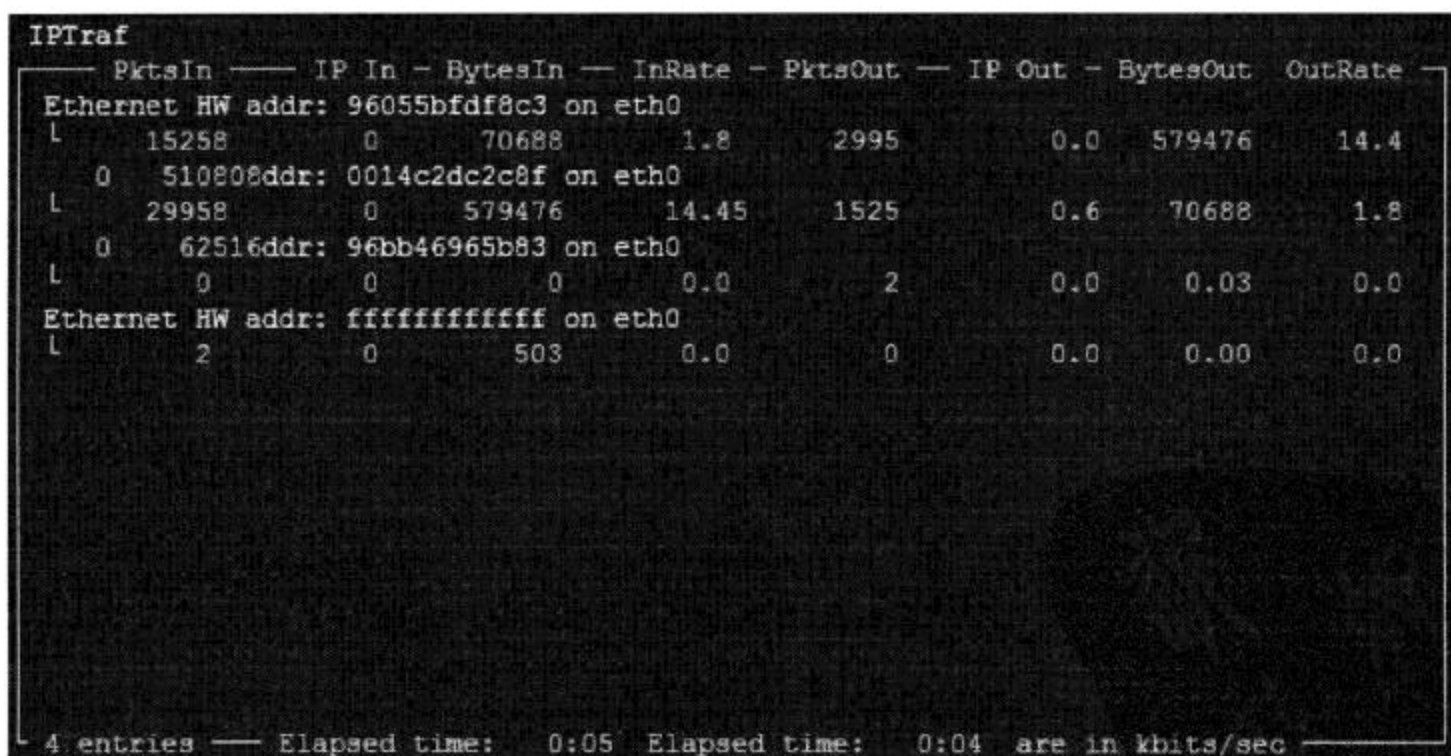
IPtraf 是 Linux 命令行下使用的网络监视工具，可以即时监看网路流量资讯，是一款网络监测和统计生成报表的工具，很实用。公司目前将它放在网关服务器（iptables 作 NAT）上，配合 NTOP 工作，效果非常好。下面将简单介绍如何在 Centos5.5 下使用 IPtraf 进行网络流量的分析。

我们如何在 Centos5.5 下安装 IPtraf 呢？

IPtraf 在 Centos5.5 x86_64 下的安装非常容易，用以下命令即可：

```
yum -y install iptraf
```

运行 IPtraf 后会产生一个字符界面的菜单，点击 x 后可以退出 IPtraf，其工作界面如图 8-14 所示。



```

IPtraf
----- PktsIn ----- IP In - BytesIn ----- InRate - PktsOut ----- IP Out - BytesOut ----- OutRate -----
Ethernet HW addr: 96055bdf8c3 on eth0
L 15258 0 70688 1.8 2995 0.0 579476 14.4
0 510808ddr: 0014c2dc2c8f on eth0
L 29958 0 579476 14.45 1525 0.6 70688 1.8
0 62516ddr: 96bb46965b83 on eth0
L 0 0 0 0.0 2 0.0 0.03 0.0
Ethernet HW addr: ffffffff on eth0
L 2 0 503 0.0 0 0.0 0.00 0.0

4 entries ----- Elapsed time: 0:05 Elapsed time: 0:04 are in kbits/sec -----
  
```

图 8-14 IPtraf 工具的工作界面

IPtraf 中各菜单的说明如下。

(1) 菜单 Configure

在这里可以对 IPtraf 进行配置，所有的修改都将保存在文件 /var/local/iptraf/Iptraf.cfg 中。

---Reverse DNS Lookups 选项：对 IP 地址反查 DNS 名，默认是关闭的。

---TCP/UDP Service Names 选项：使用服务器代替端口号，例如用 www 代替 80，默认是关闭的。

---Force promiscuous：混杂模式，此时网卡将接收所有到达的数据，不管是不是发给自己的。

---Color：终端显示彩色，当然用 Telnet、SSH 连接的除外，也就是说用不支持颜色的终端连接肯定还是没有颜色的。

---Logging：同时产生日志文件，在/var/log/Iptraf 目录下。

---Activity mode：可以选择统计单位是 kbit/sec 还是 kbyte/sec。

---Source MAC addrs in traffic monitor：选择后会显示数据包的源 MAC 地址。

(2) 菜单 Filters

在这里可以设置过滤规则，这是最有用的选项了，当你从远端连入监控机时，自己的机器与监控机会源源不断地产生 tcp 数据包，有时很令人讨厌，此时你就可以将自己的 IP 地址排除在外了。

它包括 6 个选项，分别是：Tcp、Udp、Other IP、ARP、RARP、Non-ip。我们以 TCP 为例来说明，其他选项的配置都很相似。

□ Defining a New Filter

选择 Defining a New Filter 后，会出来一个对话框，要求填入所建的当前规则的描述名，然后回车确定，按 Ctrl + x 则会取消。在接着出现的对话框里，在 Host name/IP address：的 First 里面填源地址，Second 里填目标地址，Wildcard mask 的两个框里面分别是源地址和目标地址所对应的掩码。注意，这里的地址即可以是单个地址，也可以是一个网段，如果是单个 IP，则相应的子网掩码要填成 255.255.255.255，如果是一个网段，则填写相应的子网掩码。例如，想表示 192.168.0.0，有 256 个 IP 地址的网段，则填写 192.168.0.0，子网是：255.255.255.0，其他类推，All 则用 0.0.0.0，子网也是 0.0.0.0 表示。

在 Port 栏里填入要过滤的端口号，0 表示任意端口号。Include/Exclude 栏要求填入 I 或者 E，I 表示包括，E 表示排除。填写完毕，回车确认，按 Ctrl + x 则取消。

□ Applying a Filter

我们在上一步定义的一个或多个过滤规则会存储为一个过滤列表，在没有应用之前这个表并不起作用，但我们可以选择我们应用哪些过滤规则。所有应用的规则会一直起作用，即使重新启动 IPTraf。我们可以执行 Detaching a Filter 来取消执行当前所有应用的规则。

Editing a Defined Filter：编辑一个已经存在的规则

Deleting a Defined Filter：删除一个已经定义的规则

Detaching a Filter：取消执行当前所有应用的规则

(3) 菜单 IP Traffic Monitor

IP 数据包流量实时监控窗口，注意这里会监控所有的来往数据包，包括自己的。所以，如果你使用远程终端连接的，你的监控机将会源源不断地产生数据流，因此建议在 Filters... 菜单中将自己的 IP 过滤掉，使它不产生影响。在这里可以实时地看到每一个连接的流量状态，它有两个窗口，上面的是 TCP 的连接状态，下面的窗口可以看到 UDP、ICMP、OSPF、IGRP、IGP、IGMP、GRE、ARP、RARP 的数据包。可以点击 s 键选择排序，可以按照包的数量排序，也可按照字节的大小排序。如果因为它是实时变化的而看不太清楚的话，可以在 Configure 菜单中把 Logging 功能打开，它就会在/var/log/Iptraf 目录中记录日志，以方便你在日后查看。Logging 功能打开后，当你开始监控

IPtraffic 时，程序会提示你输入 Log 文件的文件名，默认的是 ip_traffic-1.log。

在一个比较繁忙的网络里，显示的结果可能很乱，以至于你很难找到自己感兴趣的数据，这时可以使用 Filters 菜单，来过滤显示的数据。

(4) 菜单 General Interface Statistics

这里显示每个网络设备出去和进入的数据流量统计信息，包括总计、IP 包、非 IP 包、Bad IP 包和每秒的流速，单位是 kbit/sec 或 kbyte/sec，这由 Configure 菜单的 Activity 选项决定。

如果设置了 Filter 选项，这里也受到影响。

(5) 菜单 Detailed Interface Statistics

这里包括了每个网络设备的详细统计信息，很简单，不再赘述。

(6) 菜单 Statistical Breakdowns

这里提供更详细的统计信息，可以按包的大小分类来分别统计；也可以按 Tcp/Udp 的服务来分类统计，不再赘述。

(7) 菜单 LAN Station Statistics

提供每个网络地址通过本机的数据统计信息，这个对于 iptables 作为 NAT 路由器这种情况极为有用，我们可以依次分析判断到底是哪个网段的流量过大，然后我们可以通过 NTOP 定点将此 IP 抓取出来。

2. 用 NTOP 精确监控网络

如果问什么是让网络管理员最头疼的问题，恐怕大家都会回答是网络带宽匮乏了。实际情况也确实如此，随着网络应用与网络软件的越来越多，占用带宽资源的服务也越来越多。我们究竟应该怎么管理网络成为了一个非常严重的问题。BT、P2P 等软件吞噬着网络带宽，蠕虫等网络病毒也使网络应用变得枯竭。从某种意义上讲带宽就是钱，那么我们这些网络管理员该如何有效地从监视到控制公司的网络流量呢？我们可以将 NTOP 部署在 iptables 网关（即 NAT）服务器上，以便精确监控公司内部的网络流量。

为什么需要对流量进行监控呢？

作为一名普通网络用户，在浩瀚的互联网畅游之时，可能不太会有人会注意到平静的海面下其实暗流汹涌。而作为网络管理员，则需要了解各个网段的使用情形，带宽的使用率，网络问题的瓶颈发生于何处。当网络问题发生时，必须能够很快地掌握问题的发生原因，迅速定位是线路问题、网络设备问题，还是路由和防火墙的设定问题。在一个较小的网络中，一个有经验的管理者要回答这些问题并不难，但是如果其所管理的网络范围过于庞大，那么就可能需要一个有效率的网管系统来协助管理了。在业务繁忙的工作网络中，网络突然缓慢，在重要数据往来的工作时段，留给系统管理员的响应时间只有宝贵的十几分钟，甚至几分钟。如果我们不能回答网络为什么缓慢，那么很有可能因为问题严重会影响整个公司的办公效率。所以我们必须经过科学合理的计算和统计，并且在预先建立的流量分析系统中才能找到答案。

P2P (Peer-to-Peer) 是一种用于文件交换的新技术，通过 Internet 允许建立分散的、动态的、匿名的逻辑网络。P2P 为对等连接或对等网络，采用的是点对点网络技术，可应用于文件共享交换、深度搜索、分布计算等领域。它允许个体的 PC 通过 Internet 共享文件。随着 P2P 文件交换应用

的普及，ISP 在维持和增加宽带网的收益上也面临着新的挑战 and 机遇。据有关资料统计，现有的网络中有超过 70% 的带宽被 P2P 通信占据着。P2P 通信会导致异常的流量峰值，对网络资源造成意外的变形，它所带来的网络拥塞、性能下降等问题，已影响到正常的网络应用了，如 WWW、E-mail 等，缓慢的网页浏览和收发邮件速度更引起普通用户的不满。若想控制 P2P 通信，就必须对 P2P 通信进行有效的识别，然而，许多 P2P 通信使用了不同的通信技术和协议，要想使用传统的技术来识别它们非常困难。比如，许多 P2P 协议不使用固定的端口，而是动态地使用端口，包括使用一些知名服务的端口。KaZaA 工具就是一个很好的例子，它就是使用端口 80（通常是 Web 来使用）来通信的，从而穿透了传统的基于 IP 和端口的防火墙与包过滤器。所以，通过简单的基于 IP 和端口的分类技术（分析 IP 包头、IP 地址、端口号等）很难识别、跟踪或控制这类通信。过去有一段时间，有人通过监测 6881 ~ 6889 端口来识别 BT（BitTorrent），但这种做法现在早已失效——BT 已不再使用固定的 6881 ~ 6889 端口来通信了，而是动态地使用端口。随着 P2P 应用的不断增长，更多的通信协议被使用，识别和分类 P2P 的技术必须快速、简单，以适应这种技术的变化。现在，识别 P2P 通信的方法是在应用层分析数据包，看是否有某个应用协议的特征码，然后确定通信的种类。应用层分析数据包的基本方法是，如果应用层数据包的头部有“220 ftp server ready”的特征串，则可以确定是在使用 FTP 程序；如果有“HTTP/1.1 200 ok”的特征串，则确定是在使用 HTTP 传送数据。

谈到网络流量监控，相信大家都熟悉 MRTG 这个工具，但是 MRTG 也存在以下缺点：

- ☐ 使用文本式的数据库，数据不能重复使用。
- ☐ 只能按日、周、月、年来查看数据。
- ☐ 只能画两个 DS（一条线、一个块）。
- ☐ 无管理功能。
- ☐ 没有详细日志系统。
- ☐ 无法详细了解流量的具体构成。
- ☐ 只能用于 TCP/IP 网络，对于 SAN 和 iSCSI 网络流量则无能为力。
- ☐ 不能在命令行下工作。
- ☐ 过于依赖 SNMP 协议。

MRTG 基于 SNMP 协议获取信息，对于端口的流量，MRTG 能提供精确的统计，但对于 3 层以上的信息则无从得知了，而这正是 NTOP 的强项。NTOP 能够更加直观地将网络使用量的情况和每个节点计算机的网络带宽使用详细情况显示出来。NTOP 是一种网络嗅探器，嗅探器在协助监测网络数据传输、排除网络故障等方面有着不可替代的作用。它可以通过分析网络流量来确定网络上存在的各种问题，如瓶颈效应或性能下降；也可以用来判断是否有黑客正在攻击网络系统。如果怀疑网络正在遭受攻击，通过嗅探器截获的数据包可以确定正在攻击系统的是什么类型的数据包，以及它们的源头，从而及时地做出响应，或者对网络进行相应的调整，以保证网络运行的效率和安全。

通过 NTOP 网络管理员还可以很方便地确定出哪些通信量属于某个特定的网络协议、占主要通信量的是哪个主机、各次通信的目标是哪个主机、数据包发送时间、各主机间数据包传递的间隔时间等。这些信息为网管判断网络问题及优化网络性能，提供了十分宝贵的信息。NTOP 工作界面如图 8-15 所示。NTOP 提供以下一些功能：

Host	Location	Data ↓	FTP	HTTP	DNS	Telnet	MBios-IP
203.93.236.141		2.3 MBytes 48.0 %	0	240.9 KBytes	11.1 KBytes	0	0
113.57.224.3		2.1 MBytes 44.1 %	0	95.8 KBytes	0	0	0
mirrors.sohu.com		188.4 KBytes 3.8 %	0	188.4 KBytes	0	0	0
203.93.236.146		129.4 KBytes 2.6 %	0	129.4 KBytes	0	0	0
vxrp.ncast.net		38.1 KBytes 0.8 %	0	0	0	0	0
as1.hb.cnc.cn		11.1 KBytes 0.2 %	0	0	11.1 KBytes	0	0
mirror01.idc.hinet.net		10.0 KBytes 0.2 %	0	10.0 KBytes	0	0	0
msnbot-157-56-1-26.search.msn.com		6.8 KBytes 0.1 %	0	6.8 KBytes	0	0	0
king-cac44d428c [NetBIOS]		1.9 KBytes 0.0 %	0	0	0	0	1.9 KBytes
mirror.corbina.net		1.6 KBytes 0.0 %	0	1.6 KBytes	0	0	0
mirrors.ustc.edu.cn		478 0.0 %	0	478	0	0	0
mirror.bjtu.edu.cn		412 0.0 %	0	412	0	0	0
centos.hostace.ru		412 0.0 %	0	412	0	0	0
china-x937bev6g [NetBIOS]		243 0.0 %	0	0	0	0	243
ww-r4ratvadsih [NetBIOS]		243 0.0 %	0	0	0	0	243
58.221.47.240		114 0.0 %	0	0	0	0	0
58.131.136.221		114 0.0 %	0	0	0	0	0
187-35-174-158.dsl.telesp.net.br		102 0.0 %	0	0	0	0	0

图 8-15 NTOP 的工作界面

- ☐ 自动从网络中识别有用的信息。
- ☐ 将截获的数据包转换成易于识别的格式。
- ☐ 对网络环境中的通信失败进行分析。
- ☐ 探测网络环境下的通信瓶颈。
- ☐ 记录网络通信时间和过程。
- ☐ 自动识别客户端正在使用的操作系统。
- ☐ 可以在命令行和 Web 两种方式下运行。

NTOP 流量分析的要点如下：

- ☐ 连接性，也称可用性、连通性或可达性，严格说应该是网络的基本能力或属性。Internet 的出现及采用新技术而带来的生产力提高，导致对更高带宽和服务的需求。企业需要更高带宽的定制服务；而消费者则需要宽带连接和视频点播等服务；运营商必须在他们的目标市场需求与他们的业务现实之间取得平衡。这些都必须以网络的连接性能为基础和保障。
- ☐ 时延，定义了一个 IP 包穿越一个或多个网段所经历的时间。时延由固定时延和可变时延两部分组成。固定时延基本不变，由传播时延和传输时延构成；可变时延由中间路由器处理时延和排队等待时延两部分构成。
- ☐ 丢包率，是指丢失的 IP 包与所有 IP 包的比值。许多因素会导致数据包在网络上传输时被丢弃，例如数据包的大小及数据发送时链路的拥塞状况等。不同业务对丢包的敏感性不同，在多媒体业务中，丢包是导致图像质量降低和断帧的根本原因。
- ☐ 带宽，一般分为瓶颈带宽和可用带宽。瓶颈带宽是指当一条路径（通路）中没有其他背景流量时，网络能够提供的最大的吞吐量。可用带宽是指在网络路径（通路）存在背景流量

的情况下，能够提供给某个业务的最大吞吐量。在复杂的网络系统上面，不同的应用占用不同的带宽。重要的应用是否得到了最佳的带宽？它占的比例是多少？队列设置和网络优化是否生效？通过例如 MRTG 等网络流量分析会使其更加明确，并以图形 HTML 文档方式显示给用户，以非常直观的形式显示流量负载。

我们可以用 NTOP 主动分析避免异常流量。面对异常流量，我们应当建立一套分析系统，支持异常流量发现和报警，能够通过对一个时间段内历史数据的自动学习，获取包括总体网络流量水平、流量波动、流量跳变等在内的多种网络流量测度，并自动建立当前流量的置信度区间作为流量异常监测的基础。

如果自行建立主动型的网络分析系统，一般包括：测量节点、中心服务器、数据库和分析服务器。但对于中小企业来说难度较大。主动分析是借助产品化和集成程度较高的测量工具，有目的地对生产网络注入监控点，并根据测量数据流的传送情况来分析网络的性能。虽然这些监控点也会占用带宽，但和 P2P 下载所占用的可用带宽相比是微不足道的。排除病毒和封锁 P2P 之后，一般带宽占用前两名的应用是基于网页面的在线音频与在线视频。为了节约带宽，我们应该在工作时间段对其进行限制和封锁，配合行政监管人员效果可能更好。

NTOP 和 MRTG 相比它的安装配置更简单些，可以不使用 Web 服务器。目前市场上可网管型的交换机、路由器都支持 SNMP 协议，NTOP 支持简单网络管理协议所以可以进行网络流量监控。NTOP 几乎可以监测网络上的所有协议：TCP/UDP/ICMP、(R) ARP、IPX、Telnet、DLC、Decnet、DHCP-BOOTP、AppleTalk、Netbios、TCP/UDP、FTP、HTTP、DNS、Telnet、SMTP/POP/IMAP、SNMP、NNTP、NFS、X11、SSH 和基于 P2P 技术的协议，如 eDonkey、Overnet、Bittorrent、Gnutella 等。

以上部分内容来源于 <http://publish.it168.com/2006/0622/20060622001902.shtml>，特此注明。

3. NTOP 的安装及使用

前面简要介绍了为什么需要对流量进行监控，并对两款典型的带宽监控工具 NTOP 和 MRTG 进行了简要对比。下面向大家介绍 NTOP 软件的安装和使用。

我们在一台系统为 Centos5.5 x86_64、IP 为 192.168.1.104 的机器上安装 NTOP。

1) 由于 NTOP 在默认的 yum 源里是没有的，所以我们需要添加新 yum 源，这里选择 epel，它的安装步骤如下（epel 源的配置过程我们在工作中经常遇到，属于一个比较重要的知识点，推荐大家掌握）：

a) 下载与安装 EPEL 的 rpm 文件包。

由于这里是 64 位系统，所以选择安装相对应的 rpm 包。

```
rpm -ivh http://download.fedora.redhat.com/pub/epel/5/x86_64/epel-release-5-4.noarch.rpm
```

b) 导入 DAG 的 PGP Key。

```
[root@CentOS ~]# rpm -import /etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL
```

c) 设置/etc/yum.repos.d/epel.repo 文件中源的级别，添加 priority = 11（将其级别设置为较低级别，这样系统安装软件时会首先选择官方 yum 源，如果实在找不到它会选择 epel 源）。

/etc/yum.d/epel.repo 文件内容如下:

```
[epel]
name=Extra Packages for Enterprise Linux 5 - $basearch
#baseurl=http://download.fedoraproject.org/pub/epel/5/$basearch
mirrorlist=http://mirrors.fedoraproject.org/mirrorlist?repo=epel-5&arch=$basearch
failovermethod=priority
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL
priority=11
[epel-debuginfo]
name=Extra Packages for Enterprise Linux 5 - $basearch - Debug
#baseurl=http://download.fedoraproject.org/pub/epel/5/$basearch/debug
mirrorlist=http://mirrors.fedoraproject.org/mirrorlist?repo=epel-debug-5&arch=$basearch
failovermethod=priority
enabled=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL
gpgcheck=1
[epel-source]
name=Extra Packages for Enterprise Linux 5 - $basearch - Source
#baseurl=http://download.fedoraproject.org/pub/epel/5/SRPMS
mirrorlist=http://mirrors.fedoraproject.org/mirrorlist?repo=epel-source-5&arch=$basearch
failovermethod=priority
enabled=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL
gpgcheck=1
```

2) 通过 yum 安装 NTOP 极为简单, 一句话就可以了, 相对于以前版本的源码安装来说, 现在通过 yum 简化了繁琐的安装过程。命令如下所示:

```
yum -y install ntop
```

然后我们就可以输入 NTOP 命令来启动 NTOP 了, 它会要求我们设置一个 admin 密码, 配置后即可顺利启动 NTOP。

然后我们在 IE 浏览器里执行 <http://192.168.1.104:3000> (3000 是其工作端口) 就可以打开 NTOP 的工作界面, 从这里可看到所有与本机器有数据关联的信息, 如图 8-16 所示。我们比较关心的是 IP→Summary→Traffic 选项。

Host	Location	Data #	TCP	UDP	ICMP	ICMPv6	DLL	IPX	IPsec	(R)ARP	AppleTalk
192.168.1.104		455.5 KBytes 50.0 %	349.6 KBytes	8.5 KBytes	4.6 KBytes	0	0	0	0	9.5 KBytes	0
192.168.1.101		453.9 KBytes 49.9 %	348.0 KBytes	0	0	0	0	0	0	112	0
192.168.1.100		976.8 KBytes 0.1 %	885.2 KBytes	17.6 KBytes	0	0	0	0	0	9.0 KBytes	0
192.168.1.119		4.8 KBytes 0.0 %	0	0	4.6 KBytes	0	0	0	0	168	0
192.168.1.1		48 0.0 %	0	0	0	0	0	0	0	48	0

图 8-16 NTOP 的 IP 信息汇总示意图

从图中看出，IP 为 192.168.1.101 的机器在短时间内与本机 192.168.1.104 有大量的信息交互，达到了 453.9MBytes，远远高于其他机器，所以我们可以马上查看 192.168.1.101 的工作机有没有什么非法活动。这么智能的工具非常受系统管理员的欢迎，我们也可以将其作为 iptables 作 NAT 路由器时监控网络的一种手段。

8.10 TCP_wrappers 应用级防火墙的介绍和应用

了解了 iptables 防火墙后，相信大家可能会对它强大的 IP 过滤功能感到惊叹。但是在日常工作中，有时仅仅过滤 IP 是满足不了我们的工作需求的，稍后向大家介绍一种基于应用级别的防火墙，它就是强大的 TCP_wrappers。

我们为什么总说 Linux 服务器安全呢？大家可以通过下面的 Linux 系统访问控制流程看出，一个客户端访问 Linux 服务器的资源，其实也是一件不容易的事情，它需要突破层层封锁和权限控制。

Linux 系统访问控制的流程如下所示：

客户端→iptables→TCP_wrappers→Service 本身的访问控制

□ iptables：基于源 IP、目的 IP、源端口、目的端口来进行控制。

□ Tcp_wrappers：对服务的本身进行控制。

□ Service：对行为进行控制（它会结合文件和目录权限做更细致的控制）。

TCP_wrappers 是根据 /etc/hosts.allow 及 /etc/hosts.deny 这两个文件来判断用户是否能够访问服务器资源的。其实，/etc/hosts.allow 与 /etc/hosts.deny 是 /usr/sbin/tcpd 的设定档，/usr/bin/tcpd 则是用来分析进入系统的 TCP 封包的一个软件，它是由 TCP Wrappers 所提供的。那为什么叫做 TCP_wrappers 呢？其中 wrappers 有包裹的意思，也就是说，这个套件本身的功能就是在分析 TCP 网络资料封包。前面提到网络的封包资料主要是以 TCP 封包为主，这个 TCP 封包的档头至少记录了来源和目的主机的 IP 与 port，因此，借由分析 TCP 封包，就可以决定是否让这个客户端访问服务器资源。

TCP_wrappwes 的访问控制主要是以下两个文件：

/etc/hosts.allow

/etc/hosts.deny

/etc/hosts.allow 用来定义允许的访问，/etc/hosts.deny 用来定义拒绝的访问。

注意 其实，/etc/hosts.allow 是可以定义拒绝的访问的，/etc/hosts.deny 也可以做相反的工作，但我不推荐大家这样操作，为什么我们要将简单的问题复杂化呢？

现在我们来了解一下 TCP_wrappers 的访问控制判断顺序，如果客户端 IP 通过了 iptables 防火墙后想访问我们服务器的资源，这时系统会查看此客户端请求 IP 是否存在于 /etc/hosts.allow 列表里，如果存在，则接受此 IP；如果不存在，则继续比对 /etc/hosts.deny 列表，如果存在于 hosts.deny 列表里，则拒绝此 IP 请求。如果此 IP 在两个文件里都不存在，则接受此 IP 请求。其工作流程如图 8-17 所示。此流程图逻辑清晰，建议大家也以此流程图进行记忆。

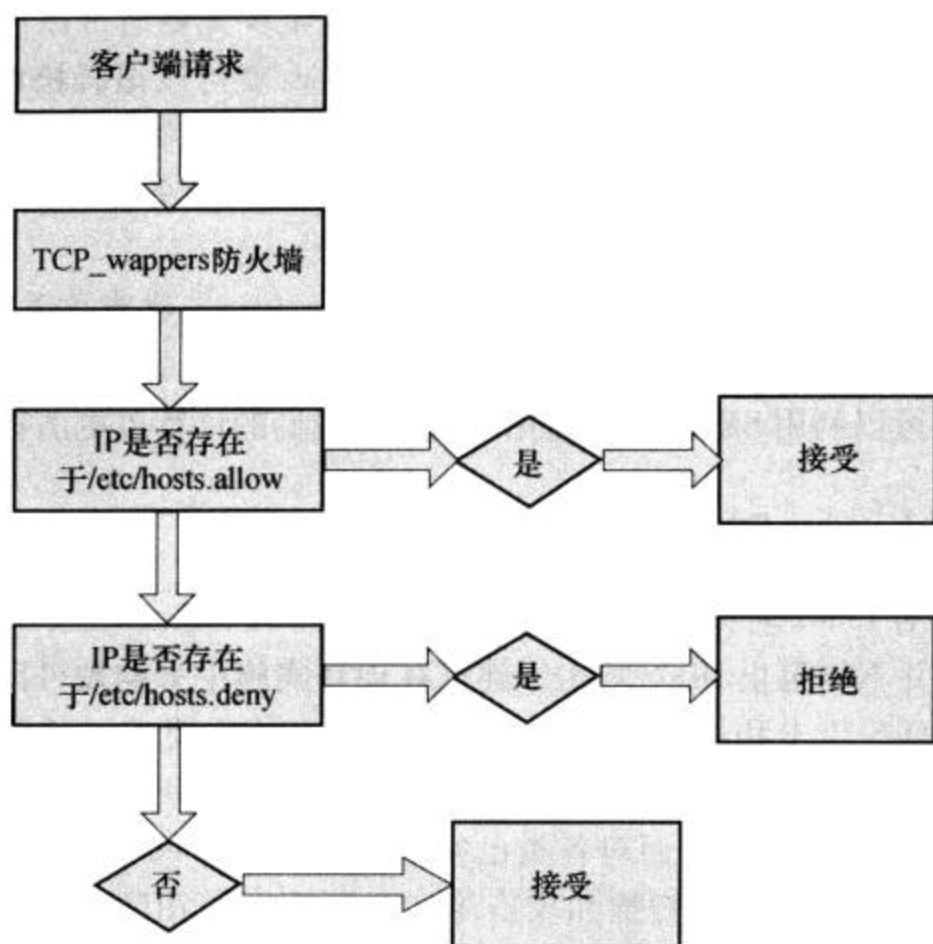


图 8-17 TCP_wappers 工作流程图

TCP_wappers 的语法格式如下：

```
<service> : <IP, domain, hostname...> : <allow|deny>
```

其语法很简单，前面接服务名，中间是：，后面是 IP 或 IP 段，最后面的 allow 或 deny 可以省略。这里也跟大家说一个小知识：在 Linux 系统里，点分十进制是除了 iptables 外，支持所有服务的语法，比如我们可以在 `/etc/hosts.allow` 里写上如下内容：

```
sshd:192.168.1.0/255.255.255.0
```

即表示此服务器接受所有来自 192.168.1.0/255.255.255.0 网段机器的 SSH 请求。下面请大家思考一个比较复杂的问题，如果一个 IP 既存在于 `/etc/hosts.allow` 里，又存在于 `/etc/hosts.deny` 里，那么这个 IP 会被服务器接受吗？例如 192.168.1.102 这个 IP 既存在于我们服务器 192.168.1.104 的 `/etc/hosts.allow` 里，又存在于 `/etc/hosts.deny` 里。其实我们用上面的流程图一判断，就可以得出结论，192.168.1.104 是允许 192.168.1.102 的机器 SSH 登录的。

那么，我们该如何利用 TCP_wappers 呢？

一些比较复杂的服务（例如 NFS 服务）在与客户端通信时会打开几个端口，如果不在配置文件里做修改，它通信的端口会与 vsftpd 服务的被动模式一样，也就是说端口都是随机的，这样我们用 iptables 来做安全策略时非常麻烦。不过，如果我们通过 TCP_wappers 来设置相应的限制策略就会非常容易，例如通过配置 `/etc/hosts.deny` 文件来限制 192.168.1.102 的机器访问本机的 NFS 资源，我们可以编辑 `/etc/hosts.deny` 文件如下：

```
vim /etc/hosts.deny
portmap:192.168.1.102
```

如果按照 iptables 防火墙的 IP + 端口防护的方法，我们至少先要通过抓包弄清楚 NFS 的端口问题才能够写 iptables 的规则，但是如果我们利用 TCP_wrappers 就可以很轻松地做好这个工作。可以说 TCP_wrappers 是 iptables 最好的补充手段之一，所以很多朋友也称其为应用级防火墙。我们在做服务器的防护工作时，可以利用 iptables + TCP_wrappers 的方法对服务器资源进行保护，这样其安全性也可以进一步增强。

到目前为止，我们已经知道了如何使用 iptables + TCP_wrappers 建造及维护一个 Linux 系统 IP 级及应用级的防火墙，它在防范外部攻击时能取得非常好的效果，但是它们防范不了来自防火墙内部的攻击。这个时候我们可以利用 SELinux 来强化内核，使我们的计算机更加安全。

8.11 工作中的 Linux 防火墙总结

以下是我在工作中对 Linux 防火墙 iptables 进行的总结：

1) iptables 防火墙并不能阻止 DDoS 攻击，建议在项目实施中采购硬件防火墙，并将其置于整个系统之前，用于防 DDoS 攻击和端口映射。如果对安全有特殊要求，可再加上应用层级的防火墙，比如天泰防火墙，其功能强大。天泰 Web 应用防火墙基于对数据报文头部和载荷完整的检测，对 Web 应用客户端输入进行验证，从而对各类已知的及新兴的 Web 应用威胁提供全方位的防护，如 SQL 注入、跨站脚本、蠕虫、黑客扫描和攻击等。天泰 Web 应用防火墙提供对目前国内比较泛滥的 DDoS 攻击的防护，针对 Web 应用进行的带宽和资源耗尽型 DDoS 攻击，它都可轻松应对。尤其针对应用层的 DDoS 攻击，提供细粒度的防护，其他优点这里不一一介绍了。

2) 在项目实施中建议关闭 Linux 服务器的 iptables 防火墙或 FreeBSD 的 ipfw，目的是更好地提高后端服务器的网络性能，方便数据流在整个业务系统内部流通。安全方面的工作由硬件防火墙来承担。

3) 我目前主要是将 iptables 用于内部作 NAT 防火墙（NATOP + IPTraf 作为流量监控），它的性能确实强悍，而且方便管理，经迅雷测试发现，公司内部的 10MB 带宽能被利用得淋漓尽致。武汉地区比较常用的软件路由是海蜘蛛软件，这个其实也是 iptables 的二次开发，即在 iptables 里加进了一些自己开发的中文模块。两年前替朋友的网吧部署路由器时，我强烈推荐的就是以 iptables 作为 NAT 路由器，事实证明，无论是管理还是整体带宽效果都很好。

4) iptables 的 L 是命令，而 -v 和 -n 只是选项，它们不能进行组合，如 -Lvn。如果要列出防火墙详细规则，可采用 iptables-nv-L 或 iptablesq-n-v-L。

5) 如果是使用远程来调试 iptables 防火墙，最好设置 Crontab 作业定时停止防火墙，以防自己被锁定，5 分钟停止一次 iptables 即可，等整个脚本完全稳定后再关闭此 Crontab 作业。

6) 如果使用默认禁止一切策略，我们在写防火墙策略时应该开放回环接口 lo（因为禁止一切也就包括了 lo 回环口），回环接口 lo 在 Linux 系统中被用来作为提供本地、基于网络服务的专用网络接口，不用通过网络接口驱动器来发送本地数据流，而是采用操作系统通过回环接口来发送，这大大提高了性能。而关闭 lo 也会带来一些莫名其妙的问题。

7) 如果是电信或双线机房托管的服务器，在没有配置前端硬件防火墙的情况下，只要不是 LVS 环境的机器，建议开启 Linux 主机的 iptables 防火墙。如果是 Windows Server 2003 主机则开启它自带的系统防火墙，并禁 ping。

8) 如果经费足够的话，最前端的硬件防火墙最好也要做成双机冗余，防止单防火墙出问题会

导致整个网站崩溃。防火墙跟人一样，总有顶不住压力的时候，如果有双机的话，网站出问题的几率要小许多。

9) 工作中我们可以利用 iptables 与 TCP_wrappers 结合的方法来对主机进行防护、如果说 iptables 是基于 IP 来过滤的话，那么 TCP_wrappers 就是一种应用级的防火墙，两者结合起来会使 Linux 系统的安全性得到进一步的提高。

在上面的 iptables 相关环节里，我主要是向大家介绍了工作中 iptables 经常涉及的部分，而 iptables 的 mangle 链和 iptables 的模块开发没有涉及，这个在平时的工作中用得少，有兴趣的朋友可以自行研究。建议大家将精力放在 iptables 如何作 NAT 路由器上，如果环境允许的朋友，可以多熟悉下硬件防火墙的性能和特点，这些在我们的工作中很有可以涉及。

8.12 Linux 系统自身的安全防护

8.12.1 SELinux 简介

SELinux (Security-Enhanced Linux) 是美国国家安全局 (NSA) 对于强制访问控制的实现，是 Linux 上最杰出的新安全子系统。NSA 是在 Linux 社区的帮助下开发了一种访问控制体系，在这种访问控制体系的限制下，进程只能访问那些在他的任务中所需要文件。SELinux 默认安装在 Centos 和 RedHat Enterprise Linux 系统上，在其他发行版上也很容易通过其安装包得到。

SELinux 是安全加强 Linux (security enhanced Linux)。一个进程是否能访问一个文件取决于发起进程的用户和文件的权限。比如：SRVD 想访问 /var/www/html/* 中的文件，我们创建的用户 andrewy 此时是否能够访问 /var/www/html/* 的文件取决于 andrewy 的权限。我们创建的文件默认路径是 644，文件夹是 755。andrewy 可以访问 fstab 和 passwd 文件，如果黑客劫持了 SRVD 进程，就可以访问一些重要的文件了，这将是一件很可怕的事。

再比如用户 cc 创建了一个文件，想让用户 guwen 访问，怎么办呢？改变其他用户组的权限？那么其他用户将都能访问，这并不是我们想要的。在现有的模型下很难进行单个用户对单个文件权限的控制，但是我们现在可以利用 SELinux 来解决这个问题。SELinux 定义了哪些用户可以访问哪些文件，就是提供了一系列的机制，来限制用户和文件的权限。

8.12.2 SELinux 的相关设置

先说说 SELinux 的相关文件，如下所示：

- 1) 主要配置文件：/etc/selinux/config (/etc/sysconfig/selinux 是它的一个软链接)。
- 2) 配置文件中的主要设定项目，以下设置可以指定 SELinux 启用的模式：

```
SELINUX=[ enforcing | permissive | disabled ]
```

说明如下：

- ☐ enforcing：强制模式，并提供限制存取机制。
- ☐ permissive：开启功能，但是提供警告模式代替限制机制。
- ☐ disabled：关闭。

以下设置指定 SELinux 使用 Policy 项目：


```
SELINUXTYPE=[ targeted |strict |mls ]
```

说明如下：

- targeted：对部分网络应用进行保护限制。
- strict：完整保护受限模式。
- mls：mls 保护受限模式，这是一种较新的安全保护模式，目前只是处于理论阶段。

3) SELinux 用于 Grub 的选项参数，这些选项参数的作用是向 kernel 传入参数指定 SELinux 相关功能。

用于 Grub 的选项参数如下所示：

```
selinux=[ 0 |1 ]
```

用于设定内核是否支持 SELinux。

getenforce 命令可以查看当前的 SELinux 模式，如下：

```
getenforce
Disabled
```

我们在生产环境下究竟如何使用 SELinux 呢？

由于我们本身对生产环境的服务器的安全权限设置得比较严苛，非系统管理员不准在线上服务器上有账户，就是系统管理员的操作也应该有记录，如果对文件或服务配置有所改动，则必须按照工作流程发邮件通知相关的负责人。由于 SELinux 的操作比较复杂，而我们的核心服务器又是置于自己公司的内部机房的，并且放置于 DMZ 区域，同时在路由器上拒绝任何 IP 连接此网段，所以我们没有使用 SELinux 功能。但有些特殊的领域（比如军事）中会涉及 SELinux。SELinux 操作和权限控制相当麻烦，有兴趣的朋友可以自行研究。

SELinux 的相关网站如下所示：

NSA 的官方 SELinux 网址 <http://www.nsa.gov/research/selinux/index.shtml>

SELinux 的 sourceforge 项目 <http://sourceforge.net/projects/selinux>

SELinux 的邮件列表 <http://www.nsa.gov/selinux/list.html>

8.13 Linux 系统安全相关的工具

在这一节里，我会向大家介绍一些常用的系统安全相关的工具，帮助大家检查 Linux 系统的文件完整性，以及系统是否感染了 Rootkit 病毒等，找出我们的系统漏洞，防止黑客入侵，进一步加强 Linux 系统安全。

这里我还是以 Centos5.5 x86_64 系统平台为背景来说明，关于 epel 源的安装方法前面的章节中已详细说明，这里假设我们已装好了 epel 源。

8.13.1 Rootkit 检测工具 Chkrootkit

Rootkit 是单个或一组软件，它针对一个或多个弱点进行获取正式权限的攻击，或者对目标主机进行其他任何类型的攻击。很多 Rootkit 不仅仅是发起一个攻击以获得 root 权限，其同时还试图掩藏和清除攻击的行为。为了达到掩盖攻击的目的，它们删除日志文件、安装特洛伊木马或采取其他的掩盖方法。就像网络中别的攻击一样，Rootkit 通常也具有特征并且会留下一些蛛丝马迹，这些都

可以用来识别出它们。我们这里有专门的软件可对 Rootkit 的踪迹和特征进行查找，如 Chkrootkit 软件。

1. Chkrootkit 的安装

Chkrootkit 目前的最新版本是 0.49，而 epel 源中的 Chkrootkit 正好就是最新版。我们可以考虑采用 yum 安装，配置好 epel 的扩展 yum 源后，Chkrootkit 的安装就是一件非常容易的事情了，可以用如下命令来进行安装：

```
yum -y install chkrootkit
```

成功安装后，再用 rpm 命令来检查，如下所示：

```
rpm -ql chkrootkit
/etc/pam.d/chkrootkit
/etc/security/console.apps/chkrootkit
/usr/bin/chkrootkit
/usr/bin/chkrootkitX
/usr/lib64/chkrootkit-0.49
/usr/lib64/chkrootkit-0.49/check_wtmpx
/usr/lib64/chkrootkit-0.49/chkdirs
/usr/lib64/chkrootkit-0.49/chklastlog
/usr/lib64/chkrootkit-0.49/chkproc
/usr/lib64/chkrootkit-0.49/chkrootkit
/usr/lib64/chkrootkit-0.49/chkutmp
/usr/lib64/chkrootkit-0.49/chkwtmp
/usr/lib64/chkrootkit-0.49/ifpromisc
/usr/lib64/chkrootkit-0.49/strings
/usr/lib64/chkrootkit-0.49/strings-static
/usr/sbin/chkrootkit
/usr/share/applications/fedora-chkrootkit.desktop
/usr/share/doc/chkrootkit-0.49
/usr/share/doc/chkrootkit-0.49/ACKNOWLEDGMENTS
/usr/share/doc/chkrootkit-0.49/COPYRIGHT
/usr/share/doc/chkrootkit-0.49/README
/usr/share/doc/chkrootkit-0.49/README.chklastlog
/usr/share/doc/chkrootkit-0.49/README.chkwtmp
/usr/share/doc/chkrootkit-0.49/README.false_positives
/usr/share/doc/chkrootkit-0.49/chkrootkit.lsm
/usr/share/pixmaps/chkrootkit.png
```

这里显示的即是成功安装 Chkrootkit 后的相关文件。

输入以下命令，确认 Chkrootkit 的版本号，命令如下：

```
chkrootkit -V
chkrootkit version 0.49
```

2. 运行 Chkrootkit

Chkrootkit 的运行也很简单，直接执行 chkrootkit 命令即可，命令如下：

```
chkrootkit
ROOTDIR is '/'
```

```

Checking 'amd'... not found
Checking 'basename'... not infected
Checking 'biff'... not found
Checking 'chfn'... not infected
Checking 'chsh'... not infected
Checking 'cron'... not infected
Checking 'crontab'... not infected
Checking 'date'... not infected
Checking 'du'... not infected
Checking 'dirname'... not infected
Checking 'echo'... not infected
Checking 'egrep'... not infected
Checking 'env'... not infected
Checking 'find'... not infected
Checking 'fingerd'... not found
Checking 'gpm'... not infected
Checking 'grep'... not infected
Checking 'hdparm'... not infected
Checking 'su'... not infected
Checking 'ifconfig'... not infected
Checking 'inetd'... not found
Checking 'inetdconf'... not found
Checking 'identd'... not found
Checking 'init'... not infected
Checking 'killall'... not infected
Checking 'ldsopreload'... not infected
Checking 'login'... not infected
Checking 'ls'... not infected
Checking 'lsof'... not infected
Checking 'mail'... not infected
Checking 'mingetty'... not infected
Checking 'netstat'... not infected
Checking 'named'... not infected
Checking 'passwd'... not infected

```

Chkrootkit 会对系统中的重要文件进行扫描，以上结果显示是正常的。正常情况下，一般没有文件感染，如果 Chkrootkit 显示有感染文件，请认真检查是否是误报，如果确认是感染了 Rootkit，请立即从网络上断开你的服务器，同时采取措施清除 Rootkit，尽管实际上这并不容易。

3. 什么时候需要运行 Chkrootkit?

我们可以在任何需要的时候运行 Chkrootkit，并不需要指定时间表。我个人比较喜欢随意、无规律地使用它。当我们察觉到计算机中或局域网内有任何可疑的行为时就运行 Chkrootkit。无论何时运行 Chkrootkit，请保证其版本为最新版本，因为新的 Rootkit 标志及其新增的功能都已经被添加到了最新的版本中了。Chkrootkit 官方地址为 <http://www.chkrootkit.org>。

8.13.2 文件系统完整性检查工具 Tripwire

完整性是计算机安全的三大要素之一（另两大要素是保密性和可靠性），完整性涉及保证数据可信并且没有以任何方式修改或损害。保持系统的完整性是安全中的另一个层次，作为系统管理员

应该对此有深切体会，当服务器遭到黑客攻击时，在多数情况下，黑客可能对系统文件等一些重要的文件进行修改。对此，我们要用 Tripwire 建立数据完整性监测系统。虽然它不能抵御黑客攻击，以及黑客对一些重要文件所做的修改，但是可以监测文件是否被修改过，以及哪些文件被修改过，从而在被攻击后有的放矢地找出解决办法。

Tripwire 的原理是将当前的系统数据状态建立成数据库，随着文件的添加、删除和修改的变化而变化，通过系统数据现状与不断更新的数据库进行比较，来判定哪些文件被添加、删除和修改过。由于初始的数据库是在 Tripwire 本体被安装、配置后建立的，所以我们务必在服务器开放前，或者操作系统刚被安装后就用 Tripwire 构建数据完整性监测系统。Tripwire 可以对要求校验的系统文件进行类似 MD5 的扫描，而生成一个唯一的标识，即“快照”(snapshot)。当这些系统文件的大小、inode 号、权限、时间等任意属性被修改后，再次运行 Tripwire，其会进行前后属性的对比，并生成相关的详细报告。

基础的文件完整性检测经常使用的一种方法就是，获得计算机上文件的校验值，并将校验值与已知值进行比较。校验值有时会用哈希值或是签名。举例来说，许多 Linux FTP 站点常有一个 md5sum 文件，在 md5sum 文件中存放的就是各文件在 Linux FTP 服务器上对应的校验值。当我们下载一个文件时，会由下载的文件得到一个校验值，如果此值与服务器上的值相匹配，就说明此文件是完整的；如果值不匹配，可能是因为下载出现错误，这样我们就应该放弃此文件重新下载，免得浪费时间进行解压等工作。md5sum 只是基础的文件完整性检查工具，高级的工具我推荐 Tripwire。

1. Tripwire 的安装过程

我已在一台测试机上通过 epel 源成功 yum 安装了 Tripwire，并检测了它的版本号，命令如下：

```
tripwire --version
Tripwire(R) 2.4.1.1 built for x86_64 - unknown - linux - gnu
```

通过命令发现它不是最新版本，需要进行版本升级。建议大家采取源码编译安装的方式。安装前记得先安装上 gcc、gcc++ 等，命令如下：

```
yum -y install gcc gcc-c++
```

我们先进入到 /usr/local/src 目录下，然后通过 wget 工具下载其最新的源码包，如下所示：

```
http://sourceforge.net/projects/tripwire/files/tripwire-src/tripwire-2.4.2-src/tripwire-2.4.2-src.tar.bz2
```

```
tar jxvf tripwire-2.4.2-srctarbz2
cd tripwire-2.4.2-src
./configure --prefix=/usr/local/tripwire && make && make install
```

安装过程中会有两个口令，我们可以设置一个比较复杂的密码，过程过程如下：

Press ENTER to view the License Agreement. ← 按回车键阅读协议

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.

59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software -- to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you Please type "accept" to indicate your acceptance of this license agreement. [do not accept] accept ← 输入"accept"同意协议
Using configuration file ./install/install.cfg
Checking for programs specified in install configuration file...
/usr/sbin/sendmail -oi -t exists. Continuing installation.
/bin/vi exists. Continuing installation.

Verifying existence of binaries...

./bin/siggen found
./bin/tripwire found
./bin/twprint found
./bin/twadmin found

This program will copy Tripwire files to the following directories:

TWBIN: /usr/local/tripwire/sbin
TWMAN: /usr/local/tripwire/man
TWPOLICY: /usr/local/tripwire/etc
TWREPORT: /usr/local/tripwire/lib/tripwire/report
TWDB: /usr/local/tripwire/lib/tripwire
TWSITEKEYDIR: /usr/local/tripwire/etc
TWLOCALKEYDIR: /usr/local/tripwire/etc
CLOBBER is false.

Continue with installation? [y/n] y ← 键入 y 继续安装

Creating directories...

/usr/loca/tripwire/sbin: already exists
/usr/loca/tripwire/etc: created
/usr/loca/tripwire/lib/tripwire/report: created
/usr/loca/tripwire/lib/tripwire: already exists
/usr/loca/tripwire/etc: already exists
/usr/loca/tripwire/etc: already exists
/usr/loca/tripwire/man: created
/usr/loca/tripwire/doc/tripwire: created

Copying files...

/usr/loca/tripwire/doc/tripwire/COPYING: copied
/usr/loca/tripwire/doc/tripwire/TRADEMARK: copied
/usr/loca/tripwire/doc/tripwire/policyguide.txt: copied


```
/usr/loca/tripwire/etc/twpol - Linux.txt: copied
```

```
-----
The Tripwire site and local passphrases are used to
sign a variety of files, such as the configuration,
policy, and database files.
```

```
Passphrases should be at least 8 characters in length
and contain both letters and numbers.
```

```
See the Tripwire manual for more information.
```

```
-----
Creating key files...
```

```
(When selecting a passphrase, keep in mind that good passphrases typically
have upper and lower case letters, digits and punctuation marks, and are
at least 8 characters in length.)
```

```
Enter the site keyfile passphrase: ← 输入"site keyfile"口令(输入后不会显示),并且记住 这个口令
```

```
Verify the site keyfile passphrase: ← 再次确认"site keyfile"口令
```

```
Generating key (this may take several minutes)...Key generation complete.
```

```
(When selecting a passphrase, keep in mind that good passphrases typically
have upper and lower case letters, digits and punctuation marks, and are
at least 8 characters in length.)
```

```
Enter the local keyfile passphrase: ← 输入"local keyfile"口令(输入后不会显示),并且记住这个口令
```

```
Verify the local keyfile passphrase: ← 再次确认"local keyfile"口令
```

```
Generating key (this may take several minutes)...Key generation complete.
```

```
-----
Generating Tripwire configuration file...
```

```
-----
Creating signed configuration file...
```

```
Please enter your site passphrase: ← 输入"site keyfile"口令(输入后不会显示)
```

```
Wrote configuration file: /usr/loca/tripwire/etc/tw.cfg
```

```
A clear-text version of the Tripwire configuration file
```

```
/usr/loca/tripwire/etc/twcfg.txt
```

```
has been preserved for your inspection. It is recommended
that you delete this file manually after you have examined it.
```

```
-----
Customizing default policy file...
```

```
-----
Creating signed policy file...
```

```
Please enter your site passphrase: ← 输入"site keyfile"口令(输入后不会显示)
```

```
Wrote policy file: /usr/loca/tripwire/etc/tw.pol
```

```
A clear-text version of the Tripwire policy file
```

```
/usr/loca/tripwire/etc/twpol.txt
```

```
has been preserved for your inspection. This implements
a minimal policy, intended only to test essential
Tripwire functionality. You should edit the policy file
to describe your system, and then use twadmin to generate
a new signed copy of the Tripwire policy.
```

```
-----
The installation succeeded.
```

```
Please refer to
```

```
for release information and to the printed user documentation
for further instructions on using Tripwire 2.4 Open Source.
```

```
make[3]: Leaving directory '/home/admin/tripwire-2.4.1.2-src'
make[2]: Leaving directory '/home/admin/tripwire-2.4.1.2-src'
make[1]: Leaving directory '/home/admin/tripwire-2.4.1.2-src'
```

安装完成。

安装完成后，我们可以检查一下 Tripwire 的版本号，命令如下：

```
tripwire --version
Open Source Tripwire(R) 2.4.1.2 built for x86_64 - unknown - linux - gnu
```

2. Tripwire 牵涉的文件

Tripwire 牵涉的文件有如下几项。

- Tripwire 的配置文件和策略文件：这是通过 `/usr/local/tripwire/etc/twcfg.txt` 文件来定义的。
- 配置文件：定义数据库、策略文件和 Tripwire 可执行文件的位置，我们可以通过更改 `/usr/local/tripwire/etc/twpol.txt` 文件来更改此项配置。
- 策略：定义检测的对象及违规时采取的行为，它对应的文件如下。
`/usr/local/tripwire/lib/tripwire/$ (HOSTNAME) .twd`
- 数据库：用于存放生成的快照。

另外，Tripwire 为了自身的安全，防止自身被篡改，也会对自身进行加密和签名处理。其中，包括以下两个密钥：

- site 密钥：用于保护策略文件和配置文件，只要使用相同策略和配置的机器，都可以使用相同的 site 密钥，它对应的文件如下。
`/usr/local/tripwire/etc/site.key`
- local 密钥：用来保护数据库和分析报告，这肯定不会重复的，它对应的文件如下。
`/usr/local/tripwire/etc/$ (HOSTNAME) -local.key`，例如我的机器是 `research.cn7788.com-local.key`

3. 编辑 twpol.txt 来控制对哪些目录进行检查

我只省略了以下几个目录，如 `/cdrom` 等，其他按照系统默认的不做任何改动，命令如下所示：

```
vim /usr/local/tripwire/etc/twpol.txt
```

将文件中的 `/cdrom`、`/floppy` 和 `/mnt` 选项添加 # 号注释掉即可，如下所示：

```
...
{
  /boot                -> $ (ReadOnly);
  #/cdrom              -> $ (Dynamic);
  #/floppy             -> $ (Dynamic);
  #/mnt                -> $ (Dynamic);
}
```

4. 初始化数据库

初始化数据库的命令如下：

```
/usr/local/tripwire/sbin/tripwire --init
Please enter your local passphrase: ← 输入"local keyfile"口令
Parsing policy file: /usr/local/tripwire/etc/tw.pol
```

```

Generating the database...
* * * Processing Unix File System * * *
The object: "/misc" is on a different file system...ignoring.
The object: "/net" is on a different file system...ignoring.
The object: "/sys" is on a different file system...ignoring.
...
Wrote database file:
/usr/local/tripwire/lib/tripwire/research.cn7788.com.twd
The database was successfully generated.

```

5. 更新数据库

当更新了 twpol.txt 后需用以下命令更新数据库:

```

/usr/local/tripwire/sbin/tripwire --update -policy --secure -mode low /usr/local/tripwire/
etc/twpol.txt
Parsing policy file: /usr/local/tripwire/etc/twpol.txt
Please enter your local passphrase: ← 输入"local keyfile"口令
Please enter your site passphrase: ← 输入"site keyfile"口令
===== Policy Update: Processing section Unix File System.
===== Step 1: Gathering information for the new policy.
The object: "/misc" is on a different file system...ignoring.
The object: "/net" is on a different file system...ignoring.
The object: "/sys" is on a different file system...ignoring.
...
===== Step 2: Updating the database with new objects.
===== Step 3: Pruning unneeded objects from the database.
Wrote policy file: /usr/local/tripwire/etc/tw.pol
Wrote database file: /usr/local/tripwire/lib/tripwire/research.cn7788.com.twd

```

6. 检查文件的异动

安装完 Tripwire 后你可以定期检查文件是否存在异动, 加上 interactive 可显示当前结果, 命令如下:

```

/usr/local/tripwire/sbin/tripwire --check - interactive
Parsing policy file: /usr/local/tripwire/etc/tw.pol
* * * Processing Unix File System * * *
Performing integrity check...
The object: "/misc" is on a different file system...ignoring.
The object: "/net" is on a different file system...ignoring.
The object: "/sys" is on a different file system...ignoring.
...

```

7. 查看报告

所有 Tripwire 的报告以 .twr 后缀保存在 lib/tripwire 目录下, 需要使用 twprint 命令来转换成文本格式, 转成文本格式以后我们就可以利用 Linux 下强大的文本编辑查看工具 (如 Sed、awk 和 Vim) 来处理了, 熟悉 Perl 的也可以用它来处理, 命令如下:

```

/usr/local/tripwire/sbin/twprint --print -report --twrfile /usr/local/tripwire/lib/trip-
wire/report/research.cn7788.com-20110603-141347.twr > /root/research.cn7788.com_20110602.txt

```

Tripwire 的使用和维护都比较简单。但要对系统的监控，关键还是需要依靠管理员定制完整的策略和检查周期，以便及时发现问题。另外，Tripwire 只能告诉你哪些文件被修改，以及修改的属性，判断和维护还是要依靠系统管理员来操作的，所以重要资料还是要做好备份及保密工作。

8.13.3 防恶意扫描软件 PortSentry

我在检查一台 Centos5.5 服务器的安全环境时，发现很多 IP 在恶意扫描此服务器的端口，本来想部署 snort 防止入侵环境的，后来发现 snort 环境的部署非常复杂，而以上的恶意扫描完全可以用 PortSentry 来实现。PortSentry 是入侵检测工具中配置最简单、效果最直接的工具之一。PortSentry 是 Abacus 工程的一个组成部分。Abacus 工程的目标是建立一个基于主机的网络入侵检测系统，可以从 <http://www.psonic.com> 得到关于 Abacus 工程的更为详细的信息。虽然 PortSentry 被 Cisco 收购后不再开发，但丝毫不影响此软件的强大功能。PortSentry 可以实时检测几乎所有类型的网络扫描，并对扫描行为做出反应。一旦发现可疑的行为，PortSentry 可以采取如下一些特定措施来加强防范：

- 给出虚假的路由信息，把所有的信息流都重定向到一个不存在的主机上。
- 自动将对服务器进行端口扫描的主机加到 TCP-wrappers 的 `/etc/hosts.deny` 文件中去。我个人比较喜欢这种方式，因为线上许多环境下并非都能打开 iptables，这个选项也是 PortSentry 默认的功能。
- 利用 Netfilter 机制，用包过滤程序，比如 iptables 和 ipchain 等，把所有非法数据包（来自对服务器进行端口扫描的主机）都过滤掉。
- 通过 `syslog()` 函数给出一个日志消息，甚至可以返回扫描者一段警告信息。

1. PortSentry 的安装

下面详细介绍 PortSentry 工具的安装和配置方法。

从 <http://sourceforge.net/projects/sentrytools/> 下载软件的最新版 `portsentry-1.2.tar.gz`，用 root 用户执行如下命令进行安装：

```
#tar zxvf portsentry-1.2.tar.gz
#cd portsentry-1.2_beta
#make
#make install
```

进行到这步时报错了，系统生成不了 PortSentry 执行文件，我们查看 Makefile 文件时发现，`make` 后面根据操作系统的不同有许多选项。

所以我们重新执行此步操作，将目录删除重新解压缩。

然后执行 `make linux`，发现系统仍然报错，如下所示：

```
SYSTYPE = linux
Making
cc -O -Wall -DLINUX -DSUPPORT_STEALTH -o ./portsentry ./portsentry.c \
./portsentry_io.c ./portsentry_util.c
./portsentry.c: In function ?.ortSentryModeTCP?.
./portsentry.c:1187: warning: pointer targets in passing argument 3 of ?.ccept?.differ in signed-
ness
./portsentry.c: In function ?.ortSentryModeUDP?.
./portsentry.c:1384: warning: pointer targets in passing argument 6 of ?.ecvfrom?.differ in sign-
```


edness

```
./portsentry.c: In function ?.sage?.
./portsentry.c:1584: error: missing terminating " character
./portsentry.c:1585: error: ?.ourceforget?.undeclared (first use in this function)
./portsentry.c:1585: error: (Each undeclared identifier is reported only once
./portsentry.c:1585: error: for each function it appears in.)
./portsentry.c:1585: error: expected ?.?.before ?.ot?
./portsentry.c:1585: error: stray ?.?.in program
./portsentry.c:1585: error: missing terminating " character
./portsentry.c:1595: error: expected ?.?.before ?.?.token
make: * * * [linux] Error 1
```

解决方法:

打开 portsentry.c 文件, 在 1590 行左右, 将带有 Copyright 1997-2003 字样的那行内容调整为一行即可, 如图 8-18 所示。

```
void
Usage (void)
{
    printf ("PortSentry - Port Scan Detector.\n");
    printf ("Copyright 1997-2003 Craig H. Rowland <craigrowland at users dot
sourceforget dot net>\n");
    printf ("Licensing restrictions apply. Please see documentation\n");
    printf ("Version: %s\n\n", VERSION);
#ifdef SUPPORT_STEALTH
    printf ("usage: portsentry [-tcp -udp -stop -stop -sudp -sudp]\n\n");
#else
    printf ("Stealth scan detection not supported on this platform\n");
    printf ("usage: portsentry [-tcp -udp]\n\n");
#endif
    printf ("*** PLEASE READ THE DOCS BEFORE USING *** \n\n");
}
```

图 8-18 方框内的那行代码应调整为一行

调整后再执行 `make linux&& make install`, PortSentry 顺利安装, 其安装路径为 `/usr/local/psionic/portsentry`。下面的内容表示成功安装了此软件:

```
Edit /usr/local/psionic/portsentry/portsentry.conf and change
your settings if you haven't already. (route, etc)
WARNING: This version and above now use a new
directory structure for storing the program
and config files (/usr/local/psionic/portsentry).
Please make sure you delete the old files when
the testing of this install is complete.
```

2. PortSentry 的配置

(1) 修改配置文件 portsentry.conf

通过 PortSentry 进行入侵检测, 首先为它定制一份需要监视的端口清单, 以及相应的阻止对策。然后启动后台进程对这些端口进行检测, 一旦发现有人扫描了这些端口, 就启动相应的对策进行阻拦。

□ 设置端口清单

下面给出 portsentry.conf 中关于端口的默认配置情况:

```
#Un-comment these if you are really anal;
#TCP_PORTS="1,7,9,11,15,70,79,80,109,110,111,119,138,139,143,512,513,514,515,540,636,1080,
1424,2000,2001,[...]
#UDP_PORTS="1,7,9,66,67,68,69,111,137,138,161,162,474,513,517,518,635,640,641,666,700,2049,
31335,27444,34555,[...]
#Use these if you just want to be aware:
TCP_PORTS="1,11,15,79,111,119,143,540,635,1080,1524,2000,5742,6667,12345,12346,20034,27665,
31337,32771,32772,[...]
UDP_PORTS="1,7,9,69,161,162,513,635,640,641,700,37444,34555,31335,32770,32771,32772,32773,
32774,31337,54321"
#Use these for juse bare-bones
#TCP_PORTS="1,11,15,110,111,143,540,635,180,1524,2000,12345,12346,20034,32771,32772,32773,
32774,49724,54320"
#UDP_PORTS="1,7,9,69,161,162,513,640,700,32770,32771,32772,32773,32774,31337,54321"
```

可以有选择地去掉前面的注释来启用默认配置，也可以根据自己的实际情况定制一份新的清单，格式和原来的一样即可。端口列表要根据具体情况而定，假如服务器为 Web 服务器，那么 Web 端口就不需要监视。反之，如果是 FTP 服务器，那么监视 Web 端口也是有必要的。

□ portentry.conf 里的相关文件

在 portentry.conf 中自动配置了许多文件，我们看一下它们都有哪些用途。

```
IGNORE_FILE="/usr/local/psionic/portentry/portentry.ignore"
```

此文件记录允许合法扫描服务器的主机地址。

```
HISTROY_FILE="/usr/lcal/psionic/portentry/portentry.history"
```

此文件中保留了以往所有入侵主机的 IP 历史记录。

```
BLOCKED_FILE="/usr/local/psionic/portentry/portentry.blocked"
```

此文件中是已经被阻止连接的主机 IP 记录。

□ 设置路由重定向

通过配置 portentry.conf 文件，可以设置一条虚拟路由记录，把数据包重定向到一个未知的主机上，使之无法获取信息。相应配置代码如下：

```
#Generic
#KILL_ROUTE="/sbin/route add $TARGET $ 333.444.555.666"
#Generic Linux
KILL_ROUTE="/sbin/route add -host $TARGET $ gw 333.444.555.666"
```

针对不同的平台有不同的路由命令，在配置文件中选择合适自己平台的命令即可。我的服务器是 Centos5.5 x86_64，以上语法适合 Linux 平台的机器。PortSentry 非常人性化，配置文件里自带各种系统对应的配置范例，我们只需要依样操作即可。

□ 我们还可以利用 Linux 中的 iptables 命令，切断攻击主机的连接。

```
KILL_ROUTE="/usr/local/sbin/iptables -I INPUT -s $TARGET $ -j DROP"
```

也可以直接把攻击者的 IP 记录到/etc/hosts.deny 文件中，利用 TCP_wrappers 保护机制来防止攻击。

```
KILL_HOSTS_DENY = "ALL: $TARGET $ # Port Sentry blocked"
```

系统默认是利用 TCP_wrappers 来切断与主机之间的连接的。

□ 定制警告信息

我们也可以定制一条警告信息，警告攻击者。不过，手册上建议不要使用该选项，因为这样做可能会暴露主机的 IDS 系统。如下所示：

```
PORT_BANNER = "*** UNAUTHORIZED ACCESS PROHIBITED *** YOUR CONNECTION ATTEMPT HAS BEEN LOGGED. GO AWAY."
```

修改完毕后，改变文件的权限，以保证其安全性。

```
chmod 600 /usr/local/psionic/port Sentry/port Sentry.conf
```

(2) 配置 port Sentry.ignore 文件

/usr/psionic/port Sentry/port Sentry.ignore 文件中设置了希望 Port Sentry 忽略的主机 IP，即允许合法扫描的主机地址，下面是配置情况：

```
#Put hosts in here you never want blocked, This includes the IP addresses
#of all local interfaces on the protected host (i.e virtual host, mult - home)
#keep 127.0.0.1 and 0.0.0.0 to keep people from playing games.
127.0.0.1/32
0.0.0.0
#Exclude all local interfaces
192.168.1.103
192.168.1.102
127.0.0.1
```

记得带上本机地址，以防万一。

修改完成后同样需要改变文件默认的权限。

```
chmod 600 /usr/local/psionic/port Sentry/port Sentry.ignore
```

3. 启动检测模式

最后介绍一下 Port Sentry 的启动检测模式。对应 TCP 和 UDP 两种协议方式，分别有 3 种启动模式，即基本、秘密和高级秘密扫描检测模式。

- port Sentry-tcp: TCP 的基本端口绑定模式
- port Sentry-udp: UDP 的基本端口绑定模式
- port Sentry-stcp: TCP 的秘密扫描检测模式
- port Sentry-sudp: UDP 的秘密扫描检测模式
- port Sentry-atcp: TCP 的高级秘密扫描检测模式
- port Sentry-audp: UDP 的高级秘密扫描检测模式

一般情况下，建议使用秘密扫描检测模式或高级秘密扫描检测模式。

使用高级秘密扫描检测模式（Advanced Stealth Scan Detection Mode），Port Sentry 会自动检查服务器上正在运行的端口，然后把端口从配置文件中移去，只监控其他的端口。这会加快对端口扫描的反应速度，并且只占用很少的 CPU 时间。这种模式非常智能，我比较喜欢用。

启动 Port Sentry 的命令如下：

```
#/usr/psionic/portsentry/portsentry -atcp
```

可以把启动命令加到“/etc/rc.d/rc.local”脚本文件中，它就会和其他后台进程一样可以随时启动、停止并查看进程状态，这样当重新启动计算机的时候 PortSentry 就会自动运行。

4. 测试

我们在 192.168.1.102 上启动 PortSentry 后，先暂时清掉 portsentry.ignore 里的文件，在另一台 192.168.1.104 的机器上启动扫描命令 `nmap -sS 192.168.1.102`，稍等片刻我们就会发现/etc/hosts.deny 里会出现 ALL: 192.168.1.104 的字样，这证明此软件配置都是生效的。

为了证明其有效性，我拿自己线上的一台 LVS 机器部署了 portsentry1.2，查看日志发现有恶意 IP 正在扫描的机器。

```
[root@localhost portsentry_beta]# tail /var/log/messages
Jun  6 13:11:07 localhost portsentry[2555]: attackalert: TCP SYN/Normal scan from host: adsl-65-9-251-89.mia.bellsouth.net/65.9.251.89 to TCP port: 80
Jun  6 13:11:07 localhost portsentry[2555]: attackalert: Host 65.9.251.89 has been blocked via wrappers with string: "ALL: 65.9.251.89"
Jun  6 13:11:07 localhost portsentry[2555]: attackalert: TCP SYN/Normal scan from host: adsl-65-9-251-89.mia.bellsouth.net/65.9.251.89 to TCP port: 80
Jun  6 13:11:07 localhost portsentry[2555]: attackalert: Host: adsl-65-9-251-89.mia.bellsouth.net/65.9.251.89 is already blocked Ignoring
Jun  6 13:11:08 localhost portsentry[2555]: attackalert: TCP SYN/Normal scan from host: adsl-65-9-251-89.mia.bellsouth.net/65.9.251.89 to TCP port: 80
Jun  6 13:11:08 localhost portsentry[2555]: attackalert: Host: adsl-65-9-251-89.mia.bellsouth.net/65.9.251.89 is already blocked Ignoring
Jun  6 13:19:57 localhost portsentry[2555]: attackalert: TCP SYN/Normal scan from host: ns38534.ovh.net/91.121.14.153 to TCP port: 80
Jun  6 13:19:57 localhost portsentry[2555]: attackalert: Host 91.121.14.153 has been blocked via wrappers with string: "ALL: 91.121.14.153"
Jun  6 13:35:44 localhost portsentry[2555]: attackalert: TCP SYN/Normal scan from host: 61.156.31.43/61.156.31.43 to TCP port: 80
Jun  6 13:35:44 localhost portsentry[2555]: attackalert: Host 61.156.31.43 has been blocked via wrappers with string: "ALL: 61.156.31.43"
```

检查了一下/etc/hosts.deny 文件，文件内容如下：

```
ALL: 113.57.224.3
ALL: 124.238.249.246
ALL: 65.9.251.89
ALL: 91.121.14.153
ALL: 61.156.31.43
```

感觉现在的无聊人士也非常多，一天到晚在公网上开着扫描器，我们的服务器应该做好相应的安全防护措施，省得浪费宝贵的带宽和 CPU 资源在这些无聊的攻击上面。

8.14 Linux 服务器基础防护篇

我们现在的许多生产服务器都是放置在 IDC 机房的，有的并没有专业的硬件防火墙保护。我们应该如何做好其基础的安全措施呢？我觉得应该从如下几方面着手：

- 首先要保证自己的 Linux 服务器的密码绝对安全，我一般将 root 密码设置为 28 位以上，而且某些重要的服务器只能有几个人知道 root 密码，这根据公司管理层的权限来设置，如果有系统管理员离职，root 密码一定要更改。用 Linux 时间长的朋友都应该知道，更改 root 密码不会影响 Linux 的 Crontab 计划任务，而 Windows Server 2003 就不一样了，如果随意更改 administrator 密码，会直接影响其计划任务的运行。如果大家有更改 Windows Server 2003 系统管理密码的需求请谨慎操作。
- 防止 SSH 暴力破解是一个老生常谈的问题，解决这个问题有许多种方法：有的朋友喜欢用 iptables 的 recent 模块来限制单位时间内 SSH 的连接数，有的用 DenyHosts 防 SSH 暴力破解工具，而我的 Linux 服务器大多数都是处于 LVS 集群环境下，我统一采用的是 SHELL 脚本保护的方法。
- 服务器的账号管理一定要严格，服务器上除了 root 账号外，系统用户越少越好，如果非要添加授权用户，请将他的登录 SHELL 设为 nologin，即此用户是没有权力登录服务器的。终止未授权用户，定期检查系统有无多余的用户都是很必要的工作。另外，像 vsftpd、Samba 及 MySQL 的账号也要严格控制，尽可能只给它们基本工作需求的权限，像 MySQL 账号，不要给任何用户 grant 权限。
- 分析系统的日志文件，寻找入侵者曾经试图入侵系统的蛛丝马迹。last 命令是另外一个可以用来查找非授权用户登录事件的工具。last 命令输入的信息来自 /var/log/wtmp。这个文件详细地记录着每个系统用户的访问活动。但是有经验的入侵者往往会删掉 /var/log/wtmp 以清除自己非法行为的证据，但是这种清除行为还是会露出蛛丝马迹：在日志文件里留下一个没有退出的操作和与之对应的登录操作（虽然在删除 wtmp 的时候登录记录没有了，但是待其退出的时候，系统还是会把它记下来）。不过高明的入侵者会用 at 或 crontab 等自己退出之后再删文件。
- 建议不定期用 `grep error /var/log/messages` 检查自己的服务器是否存在着硬件损坏的情况，由于服务器长年搁置在机房中，最容易损坏的就是硬盘和风扇，这些方面我们在进行日常维护时要注意，最好是定期巡视我们的 IDC 托管机房。
- 建议不定期使用 Chkrootkit 应用程序对 Rootkit 的踪迹和特征进行查找，从它的报告中我们可以分析出服务器否已经感染木马。
- 推荐使用 Tripwire 或 AIDE 来检查文件系统的完整性，并做好相应的分析日志。
- 停掉一些系统不必要的服务，强化内核。多关注一下服务器的内核漏洞，现在 Linux 的很多攻击都是针对内核的，保证内核版本为 2.6.9 以上（不含 2.6.9）。

8.15 如何防止入侵

事实上，我们许多非核心的服务器并未置于自己的机房内，并且有硬件防火墙保护，这个时候我们应该如何防止黑客入侵呢？本节给出了一些保证系统安全的建议，但这些建议并未涵盖全部的保证系统安全的方法，仅仅是我提出的几点意见，如下所示：

1) 系统的软件要尽可能及时更新，特别是有重大安全隐患的软件。虽然经常更新计算机系统是本节中提到的保证系统安全性方法中最容易实现的，但是这项工作却经常被忽视。计算机最容易被攻破的情况是只让其运行但却从不对其进行更新。Linux 系统及开源软件最强的性能之一就是它

们的高安全性，而这是有原因的。当一个安全问题被揭露时，开源社区会在很短的时间内提出解决方案，这为保证开源软件的高安全性提供了条件。在问题发现的同一天就找到修复方案这种情况是很常见的，甚至是以前从没有见过的安全问题也有很多都在发现的当天就被解决了，这也是我们推荐用最新的源码包构建外网 DNS 服务器的主要原因。勤更新软件是保证系统安全很重要的一方面，虽然我推荐尽可能经常地更新软件，但是我们也要密切注意关注正在更新的软件，要保证所有的更新都不会影响正常的系统运行。

2) 保证内部网的安全。很多时候为了安全，我们将核心生产服务器放置在公司内部的机房中，然后将注意力和精力放在外网的防护工作上面，于是疏忽了内部的安全性，这个时候服务器会很容易被人从内部进行破坏。正确的做法有许多种，比如我们应该将重要的生产服务器放在 DMZ 区域，跟我们的内网隔离开来，这样即使内部网络被人破坏或被人入侵，也不会影响生产服务器。

3) 尽可能最小化安装服务器和运行最少的服务。通过这么多年的实践，我们发现，安装包最小的服务器相对而言是最为稳定的。而只提供必要的基础的核心服务，也是提高我们的服务器安全稳定性的方法之一。

4) 内核的强化上面我们也要做一些工作。多关注一下服务器的内核漏洞，毕竟现在有关 Linux 的很多攻击都是针对内核的，保证内核版本为 2.6.9 以上（不含 2.6.9），推荐 Centos5.5 x86_64 以上的服务器（2.6.18-194.el5）版本。如果有相关技术支持，我们可以考虑在服务器上部署应用 SELinux。

5) 如果条件允许的话，可以在我们的网站或系统的关键位置的服务器上部署 snort。snort 是一个非常优秀的开源入侵检测软件，它集成了同类软件中最先进的技术，我们可以利用 snort 的警报找出攻击者而做出相应的防范措施。

6) 渗透测试（penetration testing）是系统安全的另一个重要方面。该测试通过使用一系列的攻击来找出系统的漏洞。顾名思义，渗透测试不是专门测试某一个或某一类攻击的，它会将所有可能的攻击拿来进行测试。这是一种找出系统漏洞的很好方法，建议大家多做一下相关的工作。

8.16 小结

本章详细向大家介绍了 iptables 和 TCP_wrappers 的语法，并演示了生产环境下的 iptables 安全脚本及 iptables 如何做企业的 NAT 路由器。接着向大家介绍了一些比较实用的安全工具，比如检查系统完整性的 Tripwire、扫描 Rootkit 的 Chkrootkit，还有命令行抓包工具 TCPDump 及图形化抓包工具 Wireshark，另外还有网络状态统计工具 iptraf 及 NTOP，以及安全扫描工具 Nmap 和 hping，并且总结了 Linux 服务器在工作中可采取的安全措施。我希望大家通过学习本章内容能对安全概念有所了解，并掌握初步的 Linux 服务器防护技巧，并且能够编写适合自己服务器的 iptables 安全脚本。



第 9 章

如何构建开源免费的企业级邮件系统

- 9.1 DNS 服务器的架设
- 9.2 电子邮件的传输过程
- 9.3 如何搭建开发邮件服务器
- 9.4 搭建 iRedmail 企业级邮件服务器
- 9.5 小结

邮件系统是企业正常运行必不可少的设施之一。微软的 Exchange2007 邮件系统很受欢迎，我也承认 Exchange2007 的性能卓越，但版权费和服务费比较昂贵，而且许多用于广告或宣传的邮件，用 Exchange2007 的效果并不是太好。那么用什么软件可以构建强大、高效、免费的邮件系统呢？现在有许多开源免费的软件能够胜任此项工作，比如 sendmail、postfix，以及 iRedmail。下面将会向大家介绍它们的具体架设方法，并以公司的邮件部署方案为例来说明 iRedmail 邮件的部署过程。

在介绍邮件系统的安装过程之前，先来了解一个重要概念：邮件服务器与 DNS 的关系。另外，有人可能以为 DNS 服务器在 Windows Server 2003 中非常容易配置，那么在 Centos 和 BSD 系统中呢？下面还会详细介绍如何在 Centos 和 BSD 系统中配置 DNS 服务器。

9.1 DNS 服务器的架设

9.1.1 邮件服务器与 DNS 的关系

事实上，目前几乎已经没有人会使用 IP 来寄信了，我们收发 E-mail 时通常都是使用“账号@主机名称”的方式来处理的，所以邮件服务器一定要有一个合法注册的主机名称。为什么呢？因为邮件服务器就是一个计算机主机，而计算机主机就是利用 TCP/IP 来进行网络数据传输的，所以需要 IP 了。但是，由于网络 IP 使用泛滥等种种原因，导致不允许直接利用主机的 IP 来寄信，因此，如果要架设邮件服务器，就必须要有合法的主机名称。

既然只要一个合法的主机名称，那么是否表示不需要架设一台 DNS 主机呢？是的，你可以这样认为！只要你拥有合法的主机名称，也就是你的主机名称在 DNS 的查询系统中拥有一个 A 标志，理论上邮件服务器就可以架设成功了。如今垃圾邮件和病毒邮件等占用了太多的带宽，导致企业要花费很多的成本来处理这些垃圾信息。为了杜绝可恶的垃圾邮件，目前的大型邮件主机供货商（ISP）都会对不明来源的邮件加以限制，也就是说想要架设一台简单且可以运作的邮件服务器越来越难了。

请大家不要忽略 DNS 的反向解析，它现在也越来越重要了，很多系统管理员配置 DNS 服务器时不喜欢反向解析，这个习惯不是太好。

对于一般的服务器来说，我们只要使用能让客户端正确找到服务器的 IP 即可建站，比如 Web 服务器。不过，由于目前收信端的邮件主机针对邮件来源的 IP 进行反向解析。如果你的网络环境是通过 ADSL 拨号取得非固定 IP 的，ISP 通常会主动以 cn7788.com 命名你的主机名，但这样的主机名称往往会被主要的大型邮件服务器（例如 hotmail 和 yahoo 等）视为垃圾邮件，所以你的邮件主机所发出的邮件可能会被丢弃。所以，如果你想要架设一台邮件服务器，请务必向你的上层 ISP 申请反向解析，否则很容易导致邮件主机所发出的邮件在网上“流浪”。

那么，邮件服务器系统到底是如何使用 DNS 的信息来进行邮件传递的呢？当一封邮件要传送出去时，邮件主机先分析邮件的目标主机的 DNS，以取得 MX 标志（注意，MX 标志可能会有多部主机哦！），然后以最优先的 MX 主机将信发送出去。我们以下面这个 DNS 范例来进一步说明。

```
cn7788.com IN MX 10 mail1.cn7788.com
cn7788.com IN MX 20 mail2.cn7788.com
cn7788.com IN A 203.93.x.x
```

假如上述的 DNS 设定是正常的，那么正常流程如下所示。

(1) 一封邮件要传送给 user@cn7788.com，根据 MX 标志最低者优先原则，它会先传送到 mail1.cn7788.com 所在主机上。

(2) 如果 mail1.cn7788.com 由于种种原因无法收到该邮件，它将以次要 MX 主机来传送，即传送到 mail2.cn7788.com 所在主机上。

(3) 如果两部 MX 主机都无法收到邮件，那么该邮件会直接以 A 的标志传送到 cn7788.com 上，也就是 203.93.x.x 的机器。

在这个过程中，你必须注意到 mail1.cn7788.com 和 mail2.cn7788.com 是可以帮 cn7788.com 转信的主机，也就是说，那两台主机通常是你公司最上游的邮件主机，并不是你随意填写的。那两台主机还需要你针对 cn7788.com 设定具体邮件的 MX 功能才行！否则你的邮件会被丢掉。

由于现在很多邮件主机都会去搜寻 MX 这个标志并根据它来判断目标邮件主机是否合法，因此你要架设邮件服务器。虽然不必自行设定 DNS 服务器，但是最好申请一个 MX 标志。此外，MX 标志一定要设置正确，否则你的邮件可能会直接被 MX 主机丢掉。如果没有上层邮件服务器，那么可以指定 MX 为自己，利用自己当 MX 主机。

根据以上所述，我们可以总结一下 MX 的作用。

MX 记录的是邮件交换记录，它指向一个邮件服务器，用于电子邮件系统发邮件时根据收信人的地址后缀来定位邮件服务器。例如，当某用户要发一封邮件给 user@mydomain.com 时，该用户的邮件系统通过 DNS 查找 mydomain.com 这个域名的 MX 记录，如果 MX 记录存在，用户计算机就将邮件发送到 MX 记录所指定的邮件服务器上。

另外，由于垃圾邮件泛滥成灾，所以 DNS 的 SPF 记录需求也产生了。那么，什么是 SPF 呢？SPF 指的是 Sender Policy Framework，中文含义是“发信者策略架构”，简称 SPF。SPF 是与 DNS 相关的一项技术，它的内容写在 DNS 的 txt 类型的记录中。MX 记录的作用是给寄信者指明某个域名的邮件服务器有哪些；SPF 的作用与 MX 相反，它向收信者表明，哪些邮件服务器是经过某个域名认可并会发送邮件的。由定义可以看出，SPF 的作用是反垃圾邮件，主要是针对那些发信人伪造域名的垃圾邮件的。例如：当 coremail 邮件服务器收到自称发件人是 spam@gmail.com 的邮件时，它到底是否真的是从 gmail.com 的邮件服务器发过来的呢？我们可以查询 gmail.com 的 SPF 记录。

市场上已经有很多邮件系统和供应商开始支持 SPF 了，比如 163.com。那么如何得到 163.com 的 SPF 值呢？在 cmd 环境中，键入 nslookup 命令行后并键入如下命令，如图 9-1 所示。

```
set type =txt
163.com
```

执行以上命令后就会得到以下的结果：

```
163.com text = "v=spf1 include:spf.163.com -all"
```

其中 "v=spf1 include:spf.163.com -all" 就是 163.com 的 SPF 值。这个数据说明了 163.com 的有效合法服务器都有哪些。



```
Microsoft Windows XP [版本 5.1.2600]
(C) 版权所有 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrator>cmd
Microsoft Windows XP [版本 5.1.2600]
(C) 版权所有 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrator>nslookup
Default Server: dns2.cs.hn.cn
Address: 202.103.96.112

> set type=txt
> 163.com
Server: dns2.cs.hn.cn
Address: 202.103.96.112

Non-authoritative answer:
163.com text =

        "v=spf1 include:spf.163.com -all"
>
```

图 9-1 用 nslookup 得出 163.com 的 SPF 值

另外，大家要注意 SPF 与 MX 的不同点。事实上，SPF 需要 DNS 以某种特定的方式来工作，也就是必须提供所谓的“反向 MX 解析”记录，这些记录是用来解析来自给定域名的邮件所对应的发送主机的。这与目前使用的 MX 记录不同，后者是用来解析给定域名所对应的接收邮件的主机的。

9.1.2 如何架设内部 DNS 服务器

本节内容的初稿我曾发表在个人的博客 (<http://hi.baidu.com/yuhongchun027>) 上供大家讨论学习，许多朋友都据此成功架设了自己的内部 DNS 服务器，并且提出了许多宝贵的改进意见。我以 Centos5.5x86_64 为平台重新整理了这部分内容。

1. 在 Centos5.5x86_64 下用 rpm 包安装 DNS 服务器

如果想用 rpm 包来依次安装，可以参照如下顺序进行，不然会发生因为依赖关系而报错的情况。rpm 包的文件罗列如下：

- ☐ bind-9.3.3-10.el5
- ☐ bind-libbind-devel-9.3.3-10.el5
- ☐ bind-sdb-9.3.3-10.el5
- ☐ bind-devel-9.3.3-10.el5
- ☐ caching-nameserver
- ☐ bind-chroot-9.3.3-10.el5
- ☐ bind-chroot (此包放最后一个安装)

如果出现以下报错信息：

```
Locating /var/named/chroot/etc/named.conf failed: [失败]
```

则将 `/usr/share/doc/bind-9.3.3/sample/etc/named.conf` 范本文件复制为 `/etc/named.conf` 即可。

使用了 chroot 后，虚拟根目录为 `/var/named/chroot`，`named.conf` 的实际位置为 `/var/named/chroot/etc`，而工作目录 `/var/named` 的实际路径为 `/var/named/chroot/var/named`。

Centos5.5 的 yum 越来越成熟和方便了，推荐大家采用 yum 安装，这样就不会发生 rpm 依赖性报错的情况。整个安装过程也很轻松，命令如下：

```
yum -y install bind-9.3*
```

系统安装完 `bind-chroot.x86_64` 软件包后，我们的实际工作路径为 `/var/named/chroot/var/named`。

建议不要用 `yum -y install bind *`，它会安装 bind97 相关的包，会导致 yum 报错，这个位置还是指明一下 bind 的版本比较合适。报错如下：

```
Error: bind97-libs conflicts with bind-libs
Error: bind97-chroot conflicts with bind-chroot
Error: bind97-devel conflicts with bind-libbind-devel
Error: bind97-utils conflicts with bind-utils
Error: bind97 conflicts with bind
Error: bind97-devel conflicts with bind-devel
```

安装完 `bind9.3.6` 后还要安装一个 `caching-nameserver` 包，不然无法生成示范文件，手写 bind 配置是件比较麻烦的事情，命令如下：

```
yum -y install caching-nameserver
```

安装完成后可以用命令来查看其安装路径及文件分布，如下所示：

```
rpm -ql bind
/etc/dbus-1/system.d/named.conf
/etc/logrotate.d/named
/etc/named.conf
/etc/rc.d/init.d/named
/etc/rndc.conf
/etc/rndc.key
/etc/sysconfig/named
/usr/sbin/bind - chroot - admin
/usr/sbin/dns - keygen
/usr/sbin/dnssec - keygen
/usr/sbin/dnssec - signzone
/usr/sbin/lwresd
/usr/sbin/named
/usr/sbin/named - bootconf
/usr/sbin/named - checkconf
/usr/sbin/named - checkzone
/usr/sbin/namedGetForwarders
/usr/sbin/namedSetForwarders
/usr/sbin/rndc
/usr/sbin/rndc - confgen
/usr/share/dbus-1/services/named.service
/usr/share/doc/bind-9.3.6
/usr/share/doc/bind-9.3.6/CHANGES
/usr/share/doc/bind-9.3.6/COPYRIGHT
/usr/share/doc/bind-9.3.6/README
/usr/share/doc/bind-9.3.6/README.DBUS
/usr/share/doc/bind-9.3.6/arm
/usr/share/doc/bind-9.3.6/arm/Bv9ARM-book.xml
/usr/share/doc/bind-9.3.6/arm/Bv9ARM.ch01.html
/usr/share/doc/bind-9.3.6/arm/Bv9ARM.ch02.html
/usr/share/doc/bind-9.3.6/arm/Bv9ARM.ch03.html
/usr/share/doc/bind-9.3.6/arm/Bv9ARM.ch04.html
/usr/share/doc/bind-9.3.6/arm/Bv9ARM.ch05.html
/usr/share/doc/bind-9.3.6/arm/Bv9ARM.ch06.html
/usr/share/doc/bind-9.3.6/arm/Bv9ARM.ch07.html
/usr/share/doc/bind-9.3.6/arm/Bv9ARM.ch08.html
/usr/share/doc/bind-9.3.6/arm/Bv9ARM.ch09.html
/usr/share/doc/bind-9.3.6/arm/Bv9ARM.html
/usr/share/doc/bind-9.3.6/arm/Bv9ARM.pdf
/usr/share/doc/bind-9.3.6/arm/Makefile
/usr/share/doc/bind-9.3.6/arm/Makefile.in
/usr/share/doc/bind-9.3.6/arm/README-SGML
/usr/share/doc/bind-9.3.6/arm/latex-fixup.pl
/usr/share/doc/bind-9.3.6/misc
/usr/share/doc/bind-9.3.6/misc/Makefile
/usr/share/doc/bind-9.3.6/misc/Makefile.in
/usr/share/doc/bind-9.3.6/misc/dnssec
/usr/share/doc/bind-9.3.6/misc/format-options.pl
```

```

/usr/share/doc/bind-9.3.6/misc/ipv6
/usr/share/doc/bind-9.3.6/misc/migration
/usr/share/doc/bind-9.3.6/misc/migration-4to9
/usr/share/doc/bind-9.3.6/misc/options
/usr/share/doc/bind-9.3.6/misc/options.edns
/usr/share/doc/bind-9.3.6/misc/rfc-compliance
/usr/share/doc/bind-9.3.6/misc/roadmap
/usr/share/doc/bind-9.3.6/misc/sdb
/usr/share/doc/bind-9.3.6/misc/sort-options.pl
/usr/share/doc/bind-9.3.6/sample
/usr/share/doc/bind-9.3.6/sample/etc
/usr/share/doc/bind-9.3.6/sample/etc/named.conf
/usr/share/doc/bind-9.3.6/sample/etc/named.rfc1912.zones
/usr/share/doc/bind-9.3.6/sample/etc/named.root.hints
/usr/share/doc/bind-9.3.6/sample/etc/rndc.conf
/usr/share/doc/bind-9.3.6/sample/var
/usr/share/doc/bind-9.3.6/sample/var/named
/usr/share/doc/bind-9.3.6/sample/var/named/data
/usr/share/doc/bind-9.3.6/sample/var/named/localdomain.zone
/usr/share/doc/bind-9.3.6/sample/var/named/localhost.zone
/usr/share/doc/bind-9.3.6/sample/var/named/my.external.zone.db
/usr/share/doc/bind-9.3.6/sample/var/named/my.internal.zone.db
/usr/share/doc/bind-9.3.6/sample/var/named/named.broadcast
/usr/share/doc/bind-9.3.6/sample/var/named/named.ip6.local
/usr/share/doc/bind-9.3.6/sample/var/named/named.local
/usr/share/doc/bind-9.3.6/sample/var/named/named.root
/usr/share/doc/bind-9.3.6/sample/var/named/named.zero
/usr/share/doc/bind-9.3.6/sample/var/named/slaves
/usr/share/doc/bind-9.3.6/sample/var/named/slaves/my.ddns.internal.zone.db
/usr/share/doc/bind-9.3.6/sample/var/named/slaves/my.slave.internal.zone.db
/usr/share/man/man5/named.conf.5.gz
/usr/share/man/man5/rndc.conf.5.gz
/usr/share/man/man8/dnssec-keygen.8.gz
/usr/share/man/man8/dnssec-signzone.8.gz
/usr/share/man/man8/lwresd.8.gz
/usr/share/man/man8/named-checkconf.8.gz
/usr/share/man/man8/named-checkzone.8.gz
/usr/share/man/man8/named.8.gz
/usr/share/man/man8/rndc-confgen.8.gz
/usr/share/man/man8/rndc.8.gz
/var/log/named.log
/var/named
/var/named/data
/var/named/slaves
/var/run/named

```

2. DNS 的配置过程

安装完 bind 包及 caching-nameserver 包后，DNS 服务器的配置工作就已经完成三分之一了，剩下的三分之二就是配置 BIND 服务了，具体步骤如下所示。

(1) 配置一个 DNS 来解析域名 ns.test.com 和 ftp.test.com, 假设对应的 IP 是 192.168.99.128。查看一下相关的配置文件, 命令如下:

```
ls -lsart /var/named/chroot/etc
total 48
8 -rw-r--r-- 1 root root 3519 Feb 27 2006 localtime
8 drwxr-x--- 5 root named 4096 Feb 23 06:35 ..
8 -rw-r----- 1 root named 955 Feb 23 06:35 named.rfc19128.zones
8 -rw-r----- 1 root named 12830 Feb 23 06:35 named.caching-nameserver.conf
8 -rw-r--r-- 1 root named 113 Jun 8 02:54 rndc.key
8 drwxr-x--- 2 root named 4096 Jun 8 03:35 .
```

(2) 这里可以看到在/var/named/chroot/etc/下的几个主要配置文件, 建议不要直接修改这些文件, 可以先复制它们然后再进行修改。复制时应该加上参数-p, 这样文件的属性均不会改变, 命令如下:

```
cp -p named.caching-nameserver.conf named.conf
```

(3) 用 named.conf 文件进行配置, 命令如下:

```
vim /var/named/chroot/etc/named.conf
```

大家可以与原文件进行比对, 有 4 个位置要改动, 即 listen-on、allow-query、match-clients、match-destinations。由于 bind 控制权限非常严格, 这里将其权限放大, 即将 localhost 改为 any, 代码变动如下:

```
options {
    listen-on port 53 { any; };
    listen-on-v6 port 53 { ::1; };
    directory "/var/named";
    dump-file "/var/named/data/cache_dump.db";
    statistics-file "/var/named/data/named_stats.txt";
    memstatistics-file "/var/named/data/named_mem_stats.txt";
    // Those options should be used carefully because they disable port
    // randomization
    // query-source port 53;
    // query-source-v6 port 53;
    allow-query { any; };
    allow-query-cache { localhost; };
};

logging {
    channel default_debug {
        file "data/named.run";
        severity dynamic;
    };
};

view localhost_resolver {
    match-clients { any; };
    match-destinations { any; };
    recursion yes;
    include "/etc/named.rfc19128.zones";
};
```

(4) 下面是 `named.rfc19128.zones` 的配置文件，其中包含了我的正向解析文件 `test.com.zone`、反向解析文件和 `192.168.99.zone` 文件，如下所示：

[illegible]

```
allow-update { none; };
};
```

注意 请大家看最后一个域的写法，它正好是反向写的，99.168.192 其实是 192.168.99 的网段，如果这地方写错了，等会儿会影响整个 DNS 的解析过程，希望大家注意。

(5) 可以看到这个配置文件里又引用了两个新的文件 test.com.zone 和 192.168.1.zone。这两个文件可以由范例文件拷贝过来，步骤如下：

```
cd /var/named/chroot/var/named/
cp -p localdomain.zone test.com.zone
cp -p named.local 192.168.99.zone
```

其中比较重要的是要加参数 p，否则很有可能会因为权限的原因无法启动 named 服务。好了，现在就可以编辑这两个文件了。先编辑反向文件，用命令 vim 192.168.99.zone 即可，代码如下：

```
$TTL      86400
@          IN      SOA      localhost. root.localhost. (
                                1997022700 ; Serial
                                28800      ; Refresh
                                14400      ; Retry
                                3600000    ; Expire
                                86400 )    ; Minimum

            IN      NS       test.com.
128        IN      PTR      ns.test.com.
128        IN      PTR      ftp.test.com.
```

有必要说明的是最后两行前面的“128”，它代表的完整 IP 为 192.168.99.128。接下来我们继续编辑正向解析文件，用命令 vim test.com.zone，如下所示：

```
$TTL      86400
@          IN      SOA      localhost root (
                                42          ; serial (d. adams)
                                3H          ; refresh
                                15M         ; retry
                                1W          ; expiry
                                1D )        ; minimum

            IN      NS       localhost
ns         IN      A         192.168.99.128
ftp        IN      A         192.168.99.128
```

其中 192.168.99.zone 是反向解析文件，而 test.com.zone 是正向解析文件。下一步就是修改 /etc/resolv.conf 文件了，这个文件大家应该了解，它其实就是指向你本机的 DNS 服务器地址，只有修改了这个文件才可以用自己的机器进行域名解析。当然，在 Windows XP 下我们可以直接修改 DNS 地址进行测试。

如果大家不想这么麻烦，这里也有一个技巧，在输入 nslookup 命令后，系统会提供交互式命令，可以用 server 192.168.99.128 来指定新的 DNS 服务器地址，命令如下：

```
nslookup
```

“>”后的内容是我们输入的，如下所示：

```
> server 192.168.99.128
Default server: 192.168.99.128
Address: 192.168.99.128#53
> 192.168.99.128
Server:      192.168.99.128
Address:     192.168.99.128#53
128.99.168.192.in-addr.arpa name = ns.test.com.
128.99.168.192.in-addr.arpa name = www.tset.com.
> www.test.com
Server:      192.168.99.128
Address:     192.168.99.128#53
Name: www.test.com
Address: 192.168.99.128
> ns.test.com
Server:      192.168.99.128
Address:     192.168.99.128#53
Name: ns.test.com
Address: 192.168.99.128
>
```

如果输入应有的正反向 query，DNS 均能显示正常的结果，则表示此 DNS 服务搭建成功了。在这个过程中，如果有文件内容要变动，也可以不重启 bind 服务，只需要在改动完成后输入 service named reload 即可。

3. 测试 DNS 配置是否生效

配置已经完成，请记得关闭 SELinux，否则其他人很有可能无法访问你的 DNS 服务器。也可以暂时先关闭一下，即用如下命令：

```
setenforce 0
```

配置了 DNS 后，还要记得开放本机的 TCP 和 UDP53 端口，命令如下：

```
iptables -A INPUT -p UDP -s --dport 53 -j ACCEPT
iptables -A INPUT -p TCP -s --dport 53 -j ACCEPT
```

如果是初次安装 DNS 服务，建议关闭 SELinux 和 iptables，以免影响测试结果。由于该 DNS 主要用于企业内部网的工作，所以 SELinux 和 iptables 可以关闭。下面测试 DNS 配置是否生效。

开启 DNS 服务，命令如下：

```
service named start
```

让 DNS 服务开机即运行，命令如下：

```
chkconfig named on
```

如果更改了 DNS 的相关文件而又不想更新 DNS 服务，可以用如下命令：

```
service named reload
```


4. 从 DNS 域名服务器应该如何配置

DNS 的主从配置是一种很常见的拓扑结构，以上是配置主域名服务器的过程，如何配置从域名服务器呢？其实配置过程是一样的，只需要改动 `named.rfc1912.zones` 文件即可，以下是变动部分：

```
zone "test.com" IN {
    type slave;
    file "slaves/test.com.zone";
    masters { 192.168.99.128; };
};
zone "99.168.192.in-addr.arpa" IN {
    type slave;
    file "slaves/192.168.99.zone";
    masters { 192.168.99.128; };
};
```

其他配置以此类推，内网架设 DNS 服务器的过程就讲到这里，我们可以继续添加记录，让它为我们提供更好的服务。

9.1.3 如何以源码方式安装公网 DNS 服务器

由于公网 DNS 是在 Internet 上提供服务的，如果有漏洞就会被别人攻击，很可能会带来毁灭性的破坏，所以其安全性我们绝对不能忽视。Centos5.5 自带的版本比较低，为了安全起见，推荐大家安装最新的 bind 源码包。

源码包可以到 bind 的官方网站 <ftp://ftp.isc.org/isc/bind9/> 下载，这里选择 bind-9.8.0-P2，日期为 2011-06-03。我找了一台公网机器，IP 为 203.93.236.141，Centos5.5 x86_64，DNS 服务器的工作目录为 `/usr/local/named`。安装过程如下所示。

1. 安装 bind-9.8.0 前的准备工作

安装 bind 前先安装 gcc 和 gcc-c++ 编译器，命令如下：

```
yum -y install gcc gcc-c++
```

下载 bind-9.8.0，放在 `/usr/local/src` 目录下，命令如下：

```
wget ftp://ftp.isc.org/isc/bind9/9.8.0-P2/bind-9.8.0-P2.tar.gz -P /usr/local/src
```

解压 bind-9.8.0-P2.tar.gz，命令如下：

```
tar zxvf bind-9.8.0-P2.tar.gz
```

进入 bind-9.8.0-P2 文件夹，命令如下：

```
cd bind-9.8.0-P2
```

创建安装目录，安装在 `/usr/local/named` 下，命令如下：

```
mkdir /usr/local/named
```

编译，指定安装目录 `/usr/local/named`，指定 man 目录，关闭 openssl 版本检查，开启多线程支持，命令如下：

```
./configure --prefix=/usr/local/named --mandir=/usr/local/share/man --enable-threads --
disable-openssl --version-check
```

接下来就是安装了，命令如下：

```
make && make install
```

如果没有报错，就表示安装成功了。成功后我们可以看一下 bind9 现在的版本号，命令如下：

```
/usr/local/named/sbin/named -v
BIND 9.8.0 - P2
```

2. 配置 bind-9.8.0 需要的文件

(1) 安装 RNDc，让其管理 bind9.6，大家应该对 service named reload 的作用印象深刻吧，在 bind9 里这个要靠 RNDc 来实现了。

进入 /usr/local/named/etc，命令如下：

```
cd /usr/local/named/etc
```

生成 rndc.conf 文件如下：

```
/usr/local/named/sbin/rndc - confgen > /usr/local/named/etc/rndc.conf
```

把 rndc.conf 中的 key 信息输出到 named.conf 中。

```
tail - n10 rndc.conf | head - n9 | sed -e s/# //g > named.conf
```

这里有一个问题请大家注意，rndc.conf 与 named.conf 的 key 值必须完全一样，而且不需要生成 rndc.key。

(2) 配置 named.conf 文件，这个文件其实就是 bind 服务的主配置文件，内容如下：

```
options {
    Directory "/usr/local/named";
    Pid-file "named.pid";
    listen-on port 53 {any;};
    Allow-query {any;};
    Dump-file "/usr/local/named/data/cache_dump.db";
    Statistics-file "/usr/local/named/data/named_stats.txt";
};

zone "." in {
    Type hint;
    File "named.root";
};

zone "localhost" in {
    Type master;
    File "localhost.zone";
};

zone "0.0.127.in-addr.arpa" in {
    Type master;
    File "localhost.rev";
};
```

```

zone "test.com" in {
    Type master;
    File "test.com.zone";
};

zone "236.93.203.in-addr.arpa" in {
    Type master;
    File "203.93.236.zone";
};

key "rndc-key" {
    algorithm hmac-md5;
    secret "OXV+irEfzfPRzteVYTqqCA==";
};

controls {
    inet 127.0.0.1 port 953
        allow { 127.0.0.1; } keys { "rndc-key"; };
};

```

(3) 在主配置文件/etc/named.conf 中定义一个根域，根域文件是/var/named 目录下的 named.root 文件。它是一个非常重要的文件，包含了 Internet 根服务器的名字和 IP 地址。当 bind 接到客户端的查询请求时，如果本地不能解释，也不能在 Cache 中找到相应的数据，就会通过根服务器进行逐级查询。

例如，当服务器收到 DNS 客户机的一个查询请求，要求查询一个不在本域的 www.example.com 域名时，如果 Cache 里没有相应的数据，DNS 服务器就会向 named.root 文件中列出的 Internet 根服务器发出请求，然后根服务器将查询交给负责域 .com 的授权名称服务器，域 .com 授权名称服务器再将请求交给负责域 example.com 的授权名称服务器进行查询，最后再把结果返回给客户机。

由于 Internet 根服务器的地址经常会发生变化，因此 named.root 也应该要随之更新。最新的根服务器列表可以从 ftp://ftp.rs.internic.net/domain/ 下载，文件名也是 named.root，我们可以用如下命令操作：

```
wget ftp://ftp.rs.internic.net/domain/named.root -P /usr/local/named
```

(4) 分别配置一下域名文件，包括 localhost.zone 正向解析文件及 localhost.rev 反向解析文件，还有 test.com.zone 正向解析文件及 236.93.203.zone 反向解析文件。

用 Vim 创建并编辑 /usr/local/named/localhost.zone 文件，内容如下：

```

$TTL 3600
$ORIGIN 127.0.0.1.
@      IN SOA localhost.root.localhost. (
        42          ;
        3H          ;
        15M         ;
        1W          ;
        3600);
IN NS  127.0.0.1
IN A   127.0.0.1

```

用 Vim 创建并编辑 /usr/local/named/localhost.rev 文件, 内容如下:

```
$TTL 3600
@ IN SOA localhost. root.localhost.(
    1; serial
    3600; refresh every hour
    900;  retry every 15 minutes
    3600000; expire 1000 hours
    3600); minimum 1 hour
IN NS localhost.
1 IN PTR localhost.
```

用 Vim 创建并编辑 /usr/local/named/203.93.236.zone 文件, 内容如下:

```
$TTL 86400
@ IN SOA localhost. root.localhost.(
    1997022700 ; Serial
    28800      ; Refresh
    14400      ; Retry
    3600000    ; Expire
    86400 )    ; Minimum

    IN NS ns.test.com.
141 IN PTR ns.test.com.
```

用 Vim 创建并编辑 /usr/local/named/test.com.zone 文件, 内容如下:

```
$TTL 86400
@ IN SOA localhost root(
    42      ; serial (d. adams)
    3H      ; refresh
    15M     ; retry
    1W      ; expiry
    1D)     ; minimum

    IN NS localhost
ns IN A 203.93.236.141
```

3. 启动 bind9 服务

下面就可以启动 bind 来测试安装是否成功, 命令如下:

```
/usr/local/named/sbin/named -gc /usr/local/named/etc/named.conf &
```

加 -gc 参数, 可以显示出启动日志, 以便进行出错排查。如果出现如下所示的字样, 则表示 bind9 服务启动成功:

```
08-Jun-2011 19:42:25.862 automatic empty zone: D.F.IP6.ARPA
08-Jun-2011 19:42:25.862 automatic empty zone: 8.E.F.IP6.ARPA
08-Jun-2011 19:42:25.863 automatic empty zone: 9.E.F.IP6.ARPA
08-Jun-2011 19:42:25.863 automatic empty zone: A.E.F.IP6.ARPA
08-Jun-2011 19:42:25.863 automatic empty zone: B.E.F.IP6.ARPA
08-Jun-2011 19:42:25.863 automatic empty zone: 8.B.D.0.1.0.0.2.IP6.ARPA
08-Jun-2011 19:42:25.866 command channel listening on 127.0.0.1#953
08-Jun-2011 19:42:25.866 ignoring config file logging statement due to -g option
```



```

08 - Jun - 2011 20:22:32.232 automatic empty zone: D.F.IP6.ARPA
08 - Jun - 2011 20:22:32.232 automatic empty zone: 8.E.F.IP6.ARPA
08 - Jun - 2011 20:22:32.232 automatic empty zone: 9.E.F.IP6.ARPA
08 - Jun - 2011 20:22:32.232 automatic empty zone: A.E.F.IP6.ARPA
08 - Jun - 2011 20:22:32.232 automatic empty zone: B.E.F.IP6.ARPA
08 - Jun - 2011 20:22:32.232 automatic empty zone: 8.B.D.0.1.0.0.2.IP6.ARPA
08 - Jun - 2011 20:22:32.235 command channel listening on 127.0.0.1#953
08 - Jun - 2011 20:22:32.236 ignoring config file logging statement due to -g option
08 - Jun - 2011 20:22:32.236 zone 0.0.127.in - addr.arpa/IN: has no NS records
08 - Jun - 2011 20:22:32.236 zone 0.0.127.in - addr.arpa/IN: not loaded due to errors.
08 - Jun - 2011 20:22:32.237 zone 236.93.203.in - addr.arpa/IN: loaded serial 1997022700
08 - Jun - 2011 20:22:32.238 test.com.zone:1: no TTL specified; using SOA MINTTL instead
08 - Jun - 2011 20:22:32.238 zone test.com/IN: loaded serial 42
08 - Jun - 2011 20:22:32.238 localhost.zone:3: ignoring out - of - zone data (127.0.0.1)
08 - Jun - 2011 20:22:32.238 zone localhost/IN: has 0 SOA records
08 - Jun - 2011 20:22:32.238 zone localhost/IN: has no NS records
08 - Jun - 2011 20:22:32.238 zone localhost/IN: not loaded due to errors.
08 - Jun - 2011 20:22:32.238 managed - keys - zone ./IN: loading from master file managed - keys.bind
failed: file not found
08 - Jun - 2011 20:22:32.239 managed - keys - zone ./IN: loaded serial 0
08 - Jun - 2011 20:22:32.239 running
08 - Jun - 2011 20:22:32.239 zone 236.93.203.in - addr.arpa/IN: sending notifies (serial
1997022700)
08 - Jun - 2011 20:22:32.239 zone test.com/IN: sending notifies (serial 42)

```

可以用 `lsof -i:53` 命令来查看，显示结果如下：

COMMAND	PID	USER	FD	TYPE	DEVICE	SIZE	NODE	NAME
named	8884	root	20u	IPv4	56777576			TCP localhost.localdomain:domain (LISTEN)
named	8884	root	21u	IPv4	56777578			TCP 203.93.236.141:domain (LISTEN)
named	8884	root	22u	IPv4	56777580			TCP 203.93.236.148:domain (LISTEN)
named	8884	root	512u	IPv4	56777575			UDP localhost.localdomain:domain
named	8884	root	513u	IPv4	56777577			UDP 203.93.236.141:domain
named	8884	root	514u	IPv4	56777579			UDP 203.93.236.148:domain

这表示 `bind9` 服务是正常启动的。这里还是选择 `nslookup` 进行测试，它的交互式查询非常方便，而且它不像 `host` 及 `dig` 命令出现在 Linux/Unix 系统里，它默认也存在于 Windows 系统里，命令如下所示：

```
nslookup
```

“>”后面的部分为我们互动输入的内容，如下所示：

```

> server localhost
Default server: localhost
Address: 127.0.0.1#53
> ns.test.com
Server:          localhost
Address:         127.0.0.1#53
Name: ns.test.com
Address: 203.93.236.141
> 203.93.236.141

```

```

Server:      localhost
Address:     127.0.0.1#53
141.236.93.203.in-addr.arpa name = ns.test.com.
>

```

如果显示以上内容,则表示此 DNS 的 ns.test.com 主机的正反向解析都是成功的,此 DNS 服务器可用于正常工作了。另外, Linux/Unix 的 dig 命令也是用来查询正反向解析的非常方便的工具之一,第1章已详细介绍了它的用法,这里就不再讲解了。

由于主从 DNS 配置是比较常用的 DNS 拓扑,因此下面也给出我的从 DNS 的 /usr/local/named/etc/named.conf 文件,有需求的朋友可以在此基础上配置从 DNS,内容如下:

```

options {
    Directory "/usr/local/named";
    Pid-file "named.pid";
    listen-on port 53 {any;};
    Allow-query {any;};
    Dump-file "/usr/local/named/data/cache_dump.db";
    Statistics-file "/usr/local/named/data/named_stats.txt";
};

key "rndc-key" {
    algorithm hmac-md5;
    secret "OXV+irEfzfPRzteVYTqqCA==";
};

controls {
    inet 127.0.0.1 port 953
        allow { 127.0.0.1; } keys { "rndc-key"; };
};

zone "." in {
    Type hint;
    File "named.root";
};

zone "localhost" in {
    Type master;
    File "localhost.zone";
};

zone "0.0.127.in-addr.arpa" in {
    Type master;
    File "localhost.rev";
};

zone "test.com" in {
    Type slave;
    masters { 203.93.236.141; };
    File "slaves/test.com.zone";
};

zone "236.93.203.in-addr.arpa" in {
    Type slave;
};

```

```

        masters { 203.93.236.141; };
File "slaves/203.93.236.zone";
};

```

9.1.4 维护 DNS 服务器应该注意的事项

1) 从 DNS 不能同步主 DNS 数据怎么办?

下面的内容出自于工作笔记, 时间为 2009-3-26。

公司的 bind9 服务器重新构架, 一台主服务器作为主 DNS, 两台从服务器作为辅助 DNS, bind9 服务器采用 9.6-P1 源码安装。下面是在从 DNS 上出现的问题:

```

Mar2616:04:17gdstnamed[18464]:client
115.207.47.199#20601:viewany:query(cache)'112.2.5.221.in -
addr.arpa/PTR/IN'denied
Mar2616:04:17gdstnamed[18464]:client
115.207.47.199#20602:viewany:query(cache)
'dx.3158.com.domain/A/IN'denied
Mar2616:04:17gdstnamed[18464]:client
115.207.47.199#20603:viewany:query(cache)
'dx.3158.com.domain/AAAA/IN'denied
Mar2616:04:17gdstnamed[18464]:client
115.207.47.199#20604:viewany:query(cache)'y163.net/A/IN'
denied
Mar2616:04:17gdstnamed[18464]:client
115.207.47.199#20605:viewany:query(cache)
'y163.net/AAAA/IN'denied
Mar2616:04:18gdstnamed[18464]:client
115.207.47.199#20606:viewany:query(cache)'112.2.5.221.in -
addr.arpa/PTR/IN'denied
Mar2616:04:18gdstnamed[18464]:client
115.207.47.199#20607:viewany:query(cache)
'dx.3158.com.domain/A/IN'denied
Mar2616:04:18gdstnamed[18464]:client
115.207.47.199#20608:viewany:query(cache)
'dx.3158.com.domain/AAAA/IN'denied
Mar2616:04:18gdstnamed[18464]:client
115.207.47.199#20609:viewany:query(cache)'y163.net/A/IN'
denied
Mar2616:04:19gdstnamed[18464]:client
115.207.47.199#20610:viewany:query(cache)
'y163.net/AAAA/IN'denied
Mar2616:04:19gdstnamed[18464]:client
115.207.47.199#20611:viewany:query(cache)'112.2.5.221.in -
addr.arpa/PTR/IN'denied
Mar2616:04:19gdstnamed[18464]:client
115.207.47.199#20612:viewany:query(cache)
'dx.3158.com.domain/A/IN'denied
Mar2616:04:19gdstnamed[18464]:client
115.207.47.199#20613:viewany:query(cache)

```



```
'dx.3158.com.domain/AAAA/IN'denied
Mar2616:04:19gdstnamed[18464]:client
115.207.47.199#20614:viewany:query(cache)'yl63.net/A/IN'
denied
Mar2616:04:20gdstnamed[18464]:client
115.207.47.199#20615:viewany:query(cache)
'yl63.net/AAAA/IN'denied
Mar2616:04:21gdstnamed[18464]:client
60.215.129.103#53455:viewany:query(cache)
'www.google.com/A/IN'denied
Mar2616:04:49gdstnamed[18464]:client
121.14.128.68#53455:viewCHINANET:query(cache)
'www.google.com/A/IN'denied
Mar2616:04:59gdstnamed[18464]:client
221.171.1.147#53455:viewCHINANET:query(cache)
'www.google.com/A/IN'denie
```

大家注意所有的 denied 字样，这里显示主从 DNS 的同步发生了错误。我查找了一下 DNS 的官方文档，发现新版中 cache 的处理机制有所改变。新版本的 bind 对 allow-query 有着不同的处理方式，官方建议新增加一个 allow-query-cache 选项，摘录如下：

```
QUOTE:allow-query Specifies which hosts are allowed to ask
ordinary DNS questions. allow-query may also
be specified in the zone statement, in which case it overrides the
options allow-query statement.
If not specified, the default is to allow queries from all hosts.
```

```
QUOTE:allow-query-cache Specifies which hosts are allowed to
get answers from the cache. The default is the
builtin acls localnets and localhost.
The way to set query access to the cache is now via allow-query-
cache. This differs from earlier
versions which used allow-query.
```

所以，在从 DNS 的 options 里添加一条相应的语句即可解决问题，更改内容如下：

```
key "rndc-key" {
    algorithm hmac-md5;
    secret "Rox3q+3f0gp8MKyQXx2zWw==";
};

controls {
    inet 127.0.0.1 port 953
        allow { localhost; } keys { "rndc-key"; };
};

options {
    version "9.8.12";
    directory "/var/named";
    pid-file "named.pid";

    allow-query { any; }; //此处为添加语句
};
```

2) 最近在维护 CDN 的 bind 服务器时, 发现了一个问题, 我的主 bind 服务器和从 bind 服务器都是电信机房的, 只有一个单 IP, 当从 bind 服务器作主从同步时, 仅仅能更新电信的 zone, 网通的则完全更新不到。查阅了相关资料后得出如下结论。

DNS 智能 view 的好处如下:

- 节省主机资源。原来可能需要一台 DNS Server 对外, 一台对内服务。现在有了 view, 只使用一台 Server 就能实现上面的功能了。
- 能够根据查询者 (外部 nameserver 或者内部 nameserver) 的 IP 作出判断, 对同一个域名或 IP 返回不同的结果。

根据当前国内的互联互通情况来看, 这可能是大家喜欢用 view 的原因。

使用上面的方式的缺点如下:

- 首先需要大量的 IP 地址, 如果你有多个 view, 则需要配置多个 IP。
- 多个 IP 可能带来路由的问题。

结合公司的 CDN 机房来看, 由于只有一个电信 IP, 所以发生问题了, 它不能实现多 view 的功能。有什么方法可以做到只用一个 IP 就可以实现多个 view 呢?

大家还记得 TSIG key 吗? rndc 等语句经常会要用到它。为什么它可以用来简化 view 的设置呢? 因为一旦使用了 TSIG, 则两台 Server 之间的通信都会用指定的 Key 来进行标识。通信双方必须具有一样的 Key, 如果 Key 不一致, 则另一方会拒绝请求。

是否可以从此点出发推广到 view 的配置上呢?

答案是可以的, 我们可以加另一条线路的 IP, 也可以用 TSIG key。其实还有一个办法, 用 Bind-DLZ 配合 MySQL 应该也可以解决这个问题 (这种情况适合对从 DNS 有时间要求的环境, 毕竟主从 DNS 同步还是需要时间的)。

3) 工作中的搭建主从 DNS 注意事项。

在维护 bind9 的主从服务器时, 希望在工作中注意以下几点:

- 如果主 DNS 和从 DNS 都是用 root 用户的, 则不需要考虑权限问题, 即 /var/named 写权限不需要更改任何地方, 比如更改为 named 或给 777 权限。
- 多使用 bind 自带的 rndc 命令, 这个命令异常方便。配置时多用 tail -f /var/log/messages, 我一般用其排障。
- 如果测试 bind 时发现出现 Non-authoritative answer, 也就是非授权的回答, 说明它来自其他 DNS 服务器或缓存。
- 从 DNS 服务器主要有两种主要用途, 一是作为主 NDNS 服务器的备份, 二是分担主 NDS 服务器的负载。区域传输 (Zone transfer) 是指从 DNS 从 master DNS 服务器中将区域数据库文件复制来的过程。关于启动区域传输的机制, 有以下三种情况:
 - a) DNS 服务器刚启动。
 - b) SOA 记录中的刷新闻隔到达。
 - c) 主 DNS 设置了主动通知从 DNS 数据有变化。

9.2 电子邮件的传输过程

在我们的日常工作中, 每天都在发送电子邮件, 比如 andrewy@cn7788.com 用户向 yuhongchun027@

163.com 的电子邮箱发送工作邮件。那么，邮件是怎么传输的呢？我们可以了解一下它的工作流程，如图 9-2 所示。

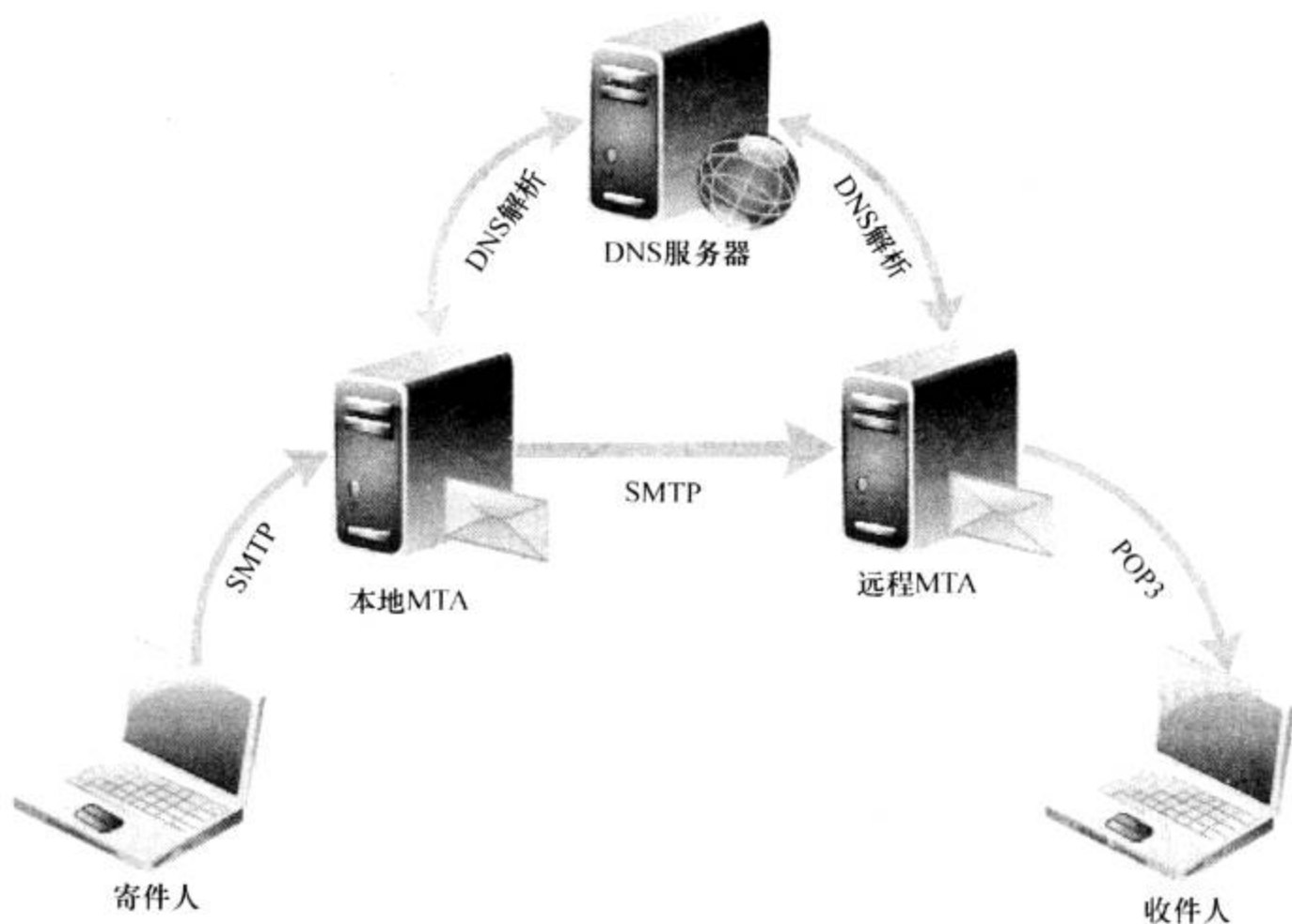


图 9-2 电子邮件传输参考模型流程

实际传输步骤如下：

1) MUA 先利用 TCP 连接端口 25，将电子邮件传送到 MTA 上，此时发件人 andrewy 必须正确定义本身与收件人的电子邮件地址，然后这些邮件会先保存在队列中。

2) 经过服务器的判断，如果收件人属于远程网络的用户，则此服务器会先向 DNS 服务器要求解析远程邮件服务器的 IP 地址。

3) 如果名称解析失败，则无法进行邮件的传递。如果成功解析远程邮件服务器的 IP 地址，则本地的邮件服务器将利用 SMTP 将邮件传送到远程（这就是邮件的转发功能）。

4) SMTP 将尝试和远程的邮件服务器连接，如果远程服务器目前无法接受邮件，则这些信件会继续停留在队列中，然后在指定的重试间隔时间内再次尝试连接，直到成功或放弃传送为止。

5) 如果传送成功，则远程 MTA 就会将此邮件交由 MDA 进行处理，并放入用户邮箱中。之后收件人 yuhongchun027 即可利用 POP 或 IMAP 软件，连接到邮件服务器上下载或读取电子邮件，而整个邮件传递过程也随之完成。

这里补充说明一下邮件的几个概念：

- MTA：邮件传送者代理人，我的理解是相当于邮局，毕竟每一个地域的邮局不一样。比如 163.com、gamil.com 或 263.net，大家看这样是不是很好理解呢？
- MDA：邮件递送代理人，我的理解为邮差。
- MUA：邮件使用者代理人，这里相当于用户。

网络中电子邮件的传递逻辑流程如图 9-3[⊖]所示。

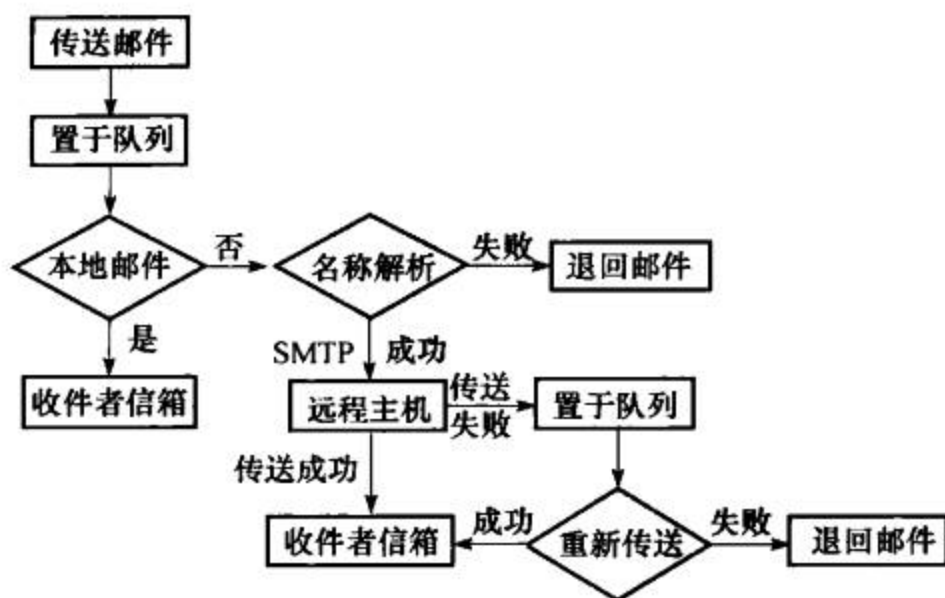


图 9-3 电子邮件传输逻辑流程图

9.3 如何搭建开发邮件服务器

很多时候，开发人员只需要简单地搭建一个邮件服务器，如果只有专门向外发送邮件的需求，这个时候可以通过 Linux/Unix 下强大的开源免费软件 Sendmail 或 Postfix 来实现。假设前面已经在 test.com 的域内配置好了一个内网的 DNS 服务器，已添加了邮件 MX 记录，如下所示：

```
test.com IN MX 10 mail.cn7788.com
```

9.3.1 搭建 Sendmail + Dovecot 邮件系统

我们现在以一台 Centos5.5 x86_64、IP 地址为 192.168.104 的机器为例，来讲解如何安装 Sendmail 邮件服务器及 Dovecot 服务器。Dovecot 是 Linux/Unix 类系统平台上的开源 IMAP 和 POP3 服务器。Dovecot 是一个比较新的软件，由 Timo Sirainen 开发，最初发布于 2002 年 7 月。作者将安全性放在第一位考虑的，所以 Dovecot 在安全性方面比较出众。

安装配置的过程如下所示。

1. 安装 Sendmail 服务器

1) 用 rpm 或 yum 命令安装 Sendmail。

在 Centos5.5 中我们可以用 system-config-packages 安装 cyrus-sasl、sendmail 及 sendmail-cf 软件包，具体包为 cyrus-sasl-2.1.22-4.i386、sendmail-8.13.8-2.el5.i386 和 sendmail-cf-8.13.8-2.el5.i386。个人推荐用 yum 安装，命令如下：

```
yum -y install sendmail*
```

此命令执行完毕后，系统会安装 sendmail 的开发包和文档，如下所示：

```
Installed:
```

⊖ 注：原图作者是李蔚译。


```
sendmail-devel.i386 0:8.13.8-8.el5      sendmail-devel.x86_64 0:8.13.8-8.el5
sendmail-doc.x86_64 0:8.13.8-8.el5
Complete!
```

安装完成后，Sendmail 的工作目录为/etc/mail。

2) 设置 local-host-names 文件，添加邮件服务器提供服务的域名，如下所示：

```
vim /etc/mail/local-host-names
```

添加的内容如下：

```
test.com
```

3) 开启 Sendmail 服务器的网络接口，如下所示：

```
vim /etc/mail/sendmail.mc
DAEMON_OPTIONS('Port = smtp,Addr = 127.0.0.1,Name = MTA') dnl
```

将 127.0.0.1 改为 0.0.0.0。

使服务器能够为主机的所有网络接口 (0.0.0.0) 提供服务。

4) 进行 Sendmail 的 SMTP 认证配置，我们可以用 Vim 编辑处理，如下所示：

```
vim /etc/mail/sendmail.mc
```

添加的内容如下所示：

```
dnl TRUST_AUTH_MECH('EXTERNAL DIGEST-MD5 CRAM-MD5 LOGIN PLAIN')dnl
dnl define('confAUTH_MECHANISMS', 'EXTERNAL GSSAPI DIGEST-MD5 CRAM-MD5 LOGIN PLAIN')dnl
```

要将这两行的 dnl 去掉，在 Sendmail 文件中 dnl 表示该行为注释行，是无效的，因此需要去除行首的 dnl 字符串来开启相应行的功能。

5) 访问控制的配置/etc/mail/access，在 Centos5 中默认了 Sendmail 服务器所在的主机用户可以任意发送邮件，而且不需要任何身份验证。注意/etc/mail/access 文件中有一行 Connect: 127.0.0.1 RELAY，RELAY 是中继的意思，如下所示：

```
Connect:localhost.localdomain      RELAY
Connect:localhost                  RELAY
Connect:127.0.0.1                  RELAY
```

默认不发生任何改动。

6) 生成 access 数据库文件，命令如下：

```
makemap hash /etc/mail/access.db < /etc/mail/access
```

7) 使用 m4 命令生成 sendmail.cf 文件，其实 sendmail.mc 是一个模板文件，如果命令报错的话，应该是 sendmail.cf 这个 rpm 没有安装，命令如下：

```
m4 sendmail.mc > sendmail.cf
```

8) 启动 Sendmail 和 saslauthd 服务，验证 Sendmail 服务，命令如下：

```
chkconfig saslauthd on
chkconfig sendmail on
```

```
service saslauthd start
service sendmail start
```

成功开启 Sendmail 服务后，我们可以用命令 `lsof` 来检查一下它是否正常启动，命令如下：

```
lsof -i:25
COMMAND  PID USER  FD  TYPE DEVICE SIZE NODE NAME
sendmail 2165 root   4u  IPv4  8101      TCP
centos5.5.cn7788.com:smtp (LISTEN)
```

9) 检查一下 Sendmail 邮件服务的 SMTP 认证是否设置成功，步骤如下：

```
telnet localhost 25
Trying 127.0.0.1...
Connected to localhost.localdomain (127.0.0.1).
Escape character is '^]'.
220 centos5.5.cn7788.com ESMTP Sendmail 8.13.8/8.13.8; Thu, 9 Jun 2011 22:11:56 +0800
ehlo localhost //此处为输入内容
250 - centos5.5.cn7788.com Hello centos5.5.cn7788.com [127.0.0.1], pleased to meet you
250 - ENHANCEDSTATUSCODES
250 - PIPELINING
250 - 8BITMIME
250 - SIZE
250 - DSN
250 - ETRN
250 - AUTH LOGIN PLAIN
250 - DELIVERBY
250 HELP
```

此时应该有 LOGIN PLAIN 的字样，即表示 SMTP 认证设置成功。

2. 配置 Dovecot 服务器

Dovecot 安装包的依赖关系比较多，我建议大家用 `yum` 来安装，命令如下：

```
yum -y install dovecot
```

将 “`#protocols = imap imaps pop3 pop3s`” 前面的 # 去掉即可。

启动 dovecot 服务，如下所示：

```
service dovecot start
```

3. 测试此邮件系统

其实到了第二步，Sendmail 邮件服务器的安装已结束了，开发人员已经可以利用此邮件服务器向外发送邮件了，安装 Dovecot 服务的作用是用来接收外部邮件的。

我们启动局域网类域名为 `test.com` 内的 DNS，在 192.168.1.104 上启动 `saslauthd`、Sendmail 及 Dovecot 服务进行测试，建议用 Foxmail 6.5 测试，其主要功能如图 9-4 所示。

`andrewy@test.com` 邮件服务器的用户账号其实是 `test.com` 域的系统用户 `andrewy`，我们可以自己建。Sendmail 邮件系统结构清晰，可作为开发专用的邮件服务器来发送邮件。如果有更强大的邮件需求请看 9.3.2 节所介绍的开源免费的 Postfix。

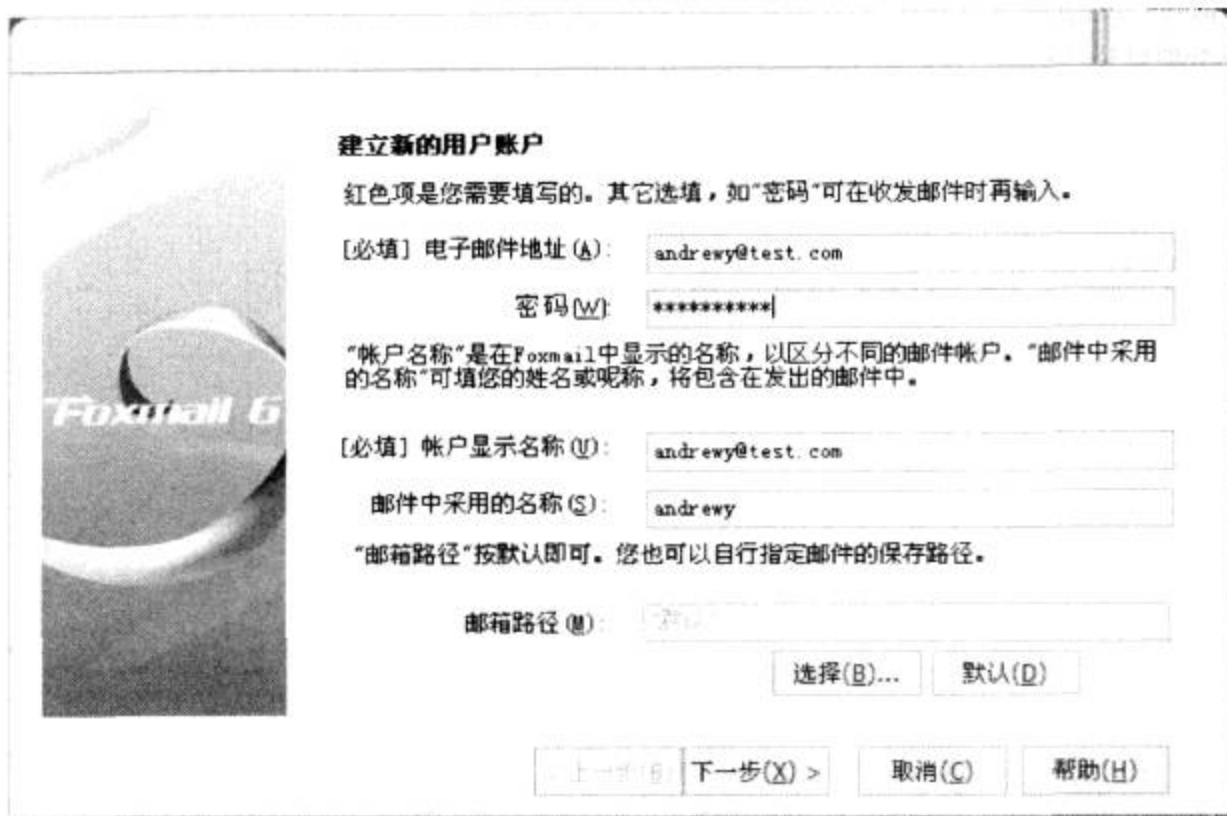


图 9-4 Foxmail 6.5 基本工作界面设置

9.3.2 搭建 Postfix + Dovecot 邮件系统

Postfix 邮件服务相对于 Sendmail 邮件服务而言，效率更优异。Postfix 邮件系统完全兼容 Sendmail，它的优点如下：

- 1) Postfix 是免费的。由于 Postfix 想要作用的范围是广大的 Internet 用户，它试图影响大多数 Internet 上的电子邮件系统用户，因此它是免费的。
- 2) Postfix 比 Sendmail 更快，大约比 Sendmail 快 3 倍。一部运行 Postfix 的台式 PC 每天可以收发上百万封邮件。
- 3) 兼容性好，Postfix 是与 Sendmail 兼容的，这样 Sendmail 用户就可以很方便地迁移到 Postfix 上。
- 4) 更健壮，Postfix 被设计成在重负荷之下仍然可以正常工作。当系统运行超出了可用的内存或磁盘空间时，Postfix 会自动减少运行进程的数目。当处理的邮件数目增长时，Postfix 运行的进程不会跟着增加。
- 5) 更灵活。Postfix 是由超过一打的小程序组成的，每个程序完成特定的功能。你可以通过配置文件设置每个程序的运行参数。
- 6) 安全性更高。Postfix 具有多层防御结构，可以有效地抵御恶意入侵者。如大多数的 Postfix 程序可以运行在较低的权限之下，不可以通过网络访问与安全性相关的本地投递程序等。

前面在域名为 test.com 的局域网环境内，已配置好了 DNS 服务器及 MX 记录，下面要在一台 Centos5.5 x86_64、IP 为 192.168.1.105 的机器上配置 Postfix + Dovecot 邮件系统，过程如下。

1. Postfix 的安装及配置过程

1) Centos5.5 跟 RedHat 的系统一样，它默认是安装及启动了 Sendmail 服务的，这里要关闭它，并配置为不随着系统启动而启动，命令如下：

```
service sendmail stop
chkconfig sendmail off
```

2) yum 安装 Postfix 软件包, 命令如下:

```
yum -y install postfix
```

安装完成后我们照例可以用命令查看其相关路径和文件, 如下所示:

```
rpm -ql postfix
```

3) 配置 Postfix 的相关文件, 它的主要配置文件为/etc/postfix/mail.cf。

```
vim /etc/postfix/main.cf
```

设置运行 Postfix 服务的邮件主机的主机名、域名, 如下所示:

```
mydomain = test.com
```

设置由本机寄出的邮件所使用的域名或主机名称, 如下所示:

```
myorigin = test.com
```

设置 Postfix 服务监听的网络接口, 如下所示:

```
inet_interfaces = all #
```

设置可接收邮件的主机名称或域名, 如下所示:

```
mydestination = $myhostname, localhost.$mydomain, localhost, $mydomain
```

设置可转发哪些网络的邮件, 如下所示:

```
mynetworks = 192.168.1.0/24,127.0.0.1/8
```

设置可转发哪些网域(当然这个也必须能由 DNS 正常解析才行)的邮件, 此选项针对上下级 MTA 而言, 区别于 Postfix 的 access, 一般选择默认选项即可, 如下所示:

```
relay_domains = $mydestination
```

4) 配置完成后我们可以检查一下 Postfix 的语法, 然后启动它。

用如下命令检查 Postfix 服务的语法:

```
/usr/sbin/postconf -n
```

启动 Postfix 邮件, 并配置为自启动服务:

```
service postfix start
chkconfig postfix on
```

2. 配置 Postfix 启用 SMTP 认证

1) 安装 cyrus-sasl 软件包, 它牵涉的依赖关系还是很多的, 命令如下:

```
yum -y install cyrus-sasl*
```

命令的结果显示如下:

```
Installed:
```



```

cyrus - sasl - devel.i386 0:2.1.22 - 5.el5_4.3
cyrus - sasl - devel.x86_64 0:2.1.22 - 5.el5_4.3
cyrus - sasl - gssapi.i386 0:2.1.22 - 5.el5_4.3
cyrus - sasl - gssapi.x86_64 0:2.1.22 - 5.el5_4.3
cyrus - sasl - ldap.i386 0:2.1.22 - 5.el5_4.3
cyrus - sasl - ldap.x86_64 0:2.1.22 - 5.el5_4.3
cyrus - sasl - md5.i386 0:2.1.22 - 5.el5_4.3
cyrus - sasl - md5.x86_64 0:2.1.22 - 5.el5_4.3
cyrus - sasl - ntlm.i386 0:2.1.22 - 5.el5_4.3
cyrus - sasl - ntlm.x86_64 0:2.1.22 - 5.el5_4.3
cyrus - sasl - sql.i386 0:2.1.22 - 5.el5_4.3
cyrus - sasl - sql.x86_64 0:2.1.22 - 5.el5_4.3
Dependency Installed:
  mysql.i386 0:5.0.77 - 4.el5_6.6      postgresql - libs.i386 0:8.1.23 - 1.el5_6.1
Dependency Updated:
  mysql.x86_64 0:5.0.77 - 4.el5_6.6    postgresql - libs.x86_64 0:8.1.23 - 1.el5_6.1
Complete!

```

我们可以用如下命令启动 saslauthd 服务，并配置成自启动服务：

```

service saslauthd start
chkconfig saslauthd on

```

2) 添加如下内容到主配置文件中，让 Postfix 启用 SMTP 认证，代码如下：

```

smtpd_sasl_auth_enable = yes
smtpd_sasl_local_domain = ''
smtpd_recipient_restrictions = permit_mynetworks, permit_sasl_authenticated, reject_unauth_
_destination
broken_sasl_auth_clients = yes
smtpd_client_restrictions = permit_sasl_authenticated
smtpd_sasl_security_options = noanonymous

```

3) 我们可以验证它是否支持 SMTP 认证，如下所示：

```

telnet localhost 25
Trying 127.0.0.1...
Connected to localhost.localdomain (127.0.0.1).
Escape character is '^]'.
220 centos5.5.cn7788.com ESMTP Postfix
ehlo localhost
250 - centos5.5.cn7788.com
250 - PIPELINING
250 - SIZE 10240000
250 - VRFY
250 - ETRN
250 - AUTH GSSAPI LOGIN PLAIN DIGEST - MD5 CRAM - MD5 NTLM
250 - AUTH=GSSAPI LOGIN PLAIN DIGEST - MD5 CRAM - MD5 NTLM
250 - ENHANCEDSTATUSCODES
250 - 8BITMIME
250 DSN

```

很明显，Postfix 服务是支持 SMTP 认证的。

3. 验证 Postfix + Dovecot 邮件系统

安装完 Dovecot 服务并启动它后，我们就可以用 Foxmail 来验证此邮件系统了，过程与前面一节中关于 Sendmail 的测试过程类似，这里就不再重复了。我在配置此邮件系统时，不小心在 `/etc/postfix/main.cf` 里将 `inet_interfaces` 参数配置了两次，一次是 `localhost`，另一次是 `all`，但只有 `localhost` 生效了，结果外部的机器连接不了此机的 25 端口，测试了很长时间（当时也很纳闷，因为这已经是第六次安装了）。请大家确保此项只有一个 `all`，另外记得关闭 SELinux 和 iptables。

初次接触邮件系统的朋友，可尝试学习一下在 Centos5.5 下搭建 Postfix + Dovecot 邮件系统，熟悉以后可以参考《POSTFIX 权威指南》的相关概念。Postfix 是一款性能优异的开源软件，我们现在能看到许多优秀的邮件系统，比如 EMOS，还有下一节将要向大家介绍的 iRedmail，都是基于 Postfix 的二次开发。正因为如此，越来越多的系统管理员或企业网管都喜欢用其搭建邮件系统，我也建议大家掌握它的基础语法。

9.4 搭建 iRedmail 企业级邮件服务器

9.4.1 iRedmail 企业级邮件服务器的介绍

iRedmail 企业级邮件服务器的搭建过程取自于我的工作文档，时间为 2009 年 2 月。我们当时组建公司企业邮件系统的初衷，主要是为了公司内部员工平时的工作交流和跟客户的沟通。初期时要求不高，只要求每日可处理 10 万封邮件，后期则达到每日要处理 100 万 ~ 200 万封邮件。另外公司成本预算不够，领导要求此邮件系统最好是免费开源的。我们测试了许多邮件系统，最终选择了 iRedmail0.4.0，现在 iRedmail 出了更新更成熟的版本 iRedmail0.0.7，大家可以关注一下它的官方网站 <http://www.iRedmail.com/>。它目前支持的平台有 RedHat Enterprise Linux、Centos、Debian、Ubuntu、OpenSUSE 及 FreeBSD。

为什么我们最终会选择 iRedmail 作为邮件系统呢？大家看看 iRedmail 的优点就明白了，如下所示：

1) iRedmail 部署起来非常方便，在保证 DNS (MX、SPF) 生效的前提下，我们 30 分钟就可以部署一个强大安全方便的邮件系统。

2) iRedmail 提供了安全的传输方式。

□ 为 Web 访问提供了 TLS/SSL 加密支持，有效保障邮件的信息安全。

□ 提供 SMTP、SMTPS 服务，为发送的邮件提供加密支持，有效保障了邮件信息的安全。

□ 提供 POP3、POP3S、IMAP、IMAPS 服务，用户可以使用 Outlook、Foxmail、Thunderbird 等邮件客户端收发邮件，并且可以选择是否使用加密传输以保证信息安全。

3) 有效地抵挡了垃圾邮件（这也是它的特色之一）。

□ 使用 SPF 技术识别邮件来源，有效防止假冒邮件。

□ 使用 DKIM 签名和校验技术。

□ 有效保证邮件顺利抵达收件人邮箱，并且收件方可以准确识别邮件，有效防止出现邮件被拒收等问题。

□ 有效验证接收到的邮件是否确实来自发件的域，有效防止垃圾邮件、假冒邮件。

□ 使用灰名单 (greylist) 技术，有效抵挡垃圾邮件。有许多系统管理员反馈，使用了灰名单技

术后，垃圾邮件减少了 20% ~ 90%。

- 还可以利用黑、白名单技术来抵挡垃圾邮件。
- 拥有强大而灵活的 HELO 抵挡技术，至少可以抵挡 20% ~ 50% 的垃圾邮件。
- 拥有强大的 spamtrap（垃圾邮件陷阱）机制。

如果通过设置一个不存在的邮件地址（例如：hr@example.com）来测试其抵制垃圾邮件的功能，所有发到这个地址的发件人都将被视为垃圾邮件发送者，会被列入黑名单。这种方法可以减少你的系统浪费在垃圾邮件识别上的时间和性能。

4) 有效的病毒防护机制，iRedmail 使用强大的开源查杀毒引擎 ClamAV 为我们的邮件提供病毒查杀服务，并且定期更新病毒库，以保证邮件安全。

5) iRedmail 提供了多种 Web Mail 界面可进行选择，它提供基于 AJAX 技术的 Web Mail 程序，易于操作，性能优异，速度快，也为你的移动办公提供了极大的便利。我们也可以选择其他 Web Mail 程序，比如 ExtMail、Horde WebMail 和 SquirrelMail（即松鼠邮）。

6) 灵活的用户策略控制。我们设置某些用户只能发送邮件到本域和子域中，还可以设置某些邮件地址只能接收来自本域用户的邮件。

- 灵活的邮件收发频率控制。例如，限制某用户在一分钟内最多只能发送或接收 5 封邮件。
- 灵活的邮件收发总容量控制。例如，限制某用户在一分钟内最多只能发送或接收 10MB 的邮件。

7) 灵活的邮件收发控制。我们可以实现以下策略：某些用户只能收或只能发，或者限制某些用户不能使用 POP3 或 IMAP 服务。

8) 强大的邮件系统管理功能。iRedmail 可以无限制地虚拟域和虚拟用户，并且还提供 Web 界面的管理后台。另外，Postfix 上的虚拟域、虚拟用户、别名功能，在 iRedmail 上均可以实现。

9.4.2 在 Centos5.2 x86_64 上安装 iRedmail0.4.0

2009 年 3 月 17 日我正式在电信机房服务器 IP 为 211.147.x.x、域名 mail.cn7789.com 的机器上部署了 iRedmail0.4.0，系统采用的是当时最新的 Centos5.2 x86_64。另外，我选择了浪潮英信 NP110D2 服务器作为我们的邮件服务器，在当时它的性价比非常高，它采用了塔式的结构，标配是 1 个奔腾 4 的处理器，核心频率 3.00GHz，前端总线 533MHz，双核共享 1 * 2M 二级缓存，我在它标准内存配置（512MB DDR2 内存）的基础上将内存增加至 4GB。在存储方面，我用的是 6 块 SATA500G 硬件做的 RAID1 + 0。

注意 这里所牵涉的具体公司和 IP 名以及邮件的域名，包括后面将牵涉的数据库密码，我均做了无害处理。

其完整步骤如下所示。

1. Centos5.2 x86_64 的系统安装组件

Centos5.2 x86_64 的系统安装组件如下所示：

- 桌面 (Y)
- 编辑器 (Y)
- 开发工具 (Y)

- ☐ 基本 (Y)
- ☐ 管理工具 (Y)
- ☐ 系统工具 (Y)
- ☐ 禁用 SELinux
- ☐ 开启 iptables

禁用 SELinux 是由于我们的服务器安全级别没达到军事级别。另外，此服务器是置于没有硬件保护的电信 IDC 机房的，所以开启 iptables 是必要的。

2. 系统内核优化

编辑内核配置文件/etc/sysctl.conf，如下所示：

```
vim /etc/sysctl.conf
```

添加的内容如下所示：

```
net.ipv4.tcp_fin_timeout = 30
net.ipv4.tcp_keepalive_time = 300
net.ipv4.tcp_tw_reuse = 1
net.ipv4.tcp_tw_recycle = 1
net.ipv4.ip_local_port_range = 5000 65000
```

修改 kernel.shmmax 项，如下所示：

```
kernel.shmmax = 134217728
```

然后不需要重启系统使设置生效，命令如下：

```
/sbin/sysctl -p
```

3. 配置额外的 yum 源，确定安装顺利

yum-fastestmirror 是 yum 的一个插件，它可以自动为你查找一个相对最快的源，安装它，命令如下：

```
yum -y install yum-fastestmirror
```

添加 iRedmail 的更新源。

```
vim /etc/yum.repos.d/CentOS-Base.repo
```

添加以下内容到此文件中：

```
[iRedmail]
name=Yum repo generated by iRedmail: http://iRedmail.googlecode.com/
baseurl=http://www.iRedmail.org/yum/rpms/5 # (这个地址我之前填错了,害得我重新安装了好几次 iRedmail)
enabled=1
gpgcheck=0
#启用 centosplus
#additional packages that extend functionality of existing
packages
[centosplus]
name=CentOS - $releasever - Plus
mirrorlist=http://mirrorlist.centos.org/?
```



```
release=$releasever&arch=$basearch&repo=centosplus
#baseurl=http://mirror.centos.org/centos/$releasever/centosplus/$basearch/
gpgcheck=1
enabled=1
gpgkey=http://mirror.centos.org/centos/RPM-GPG-KEY-CentOS-5
```

4. 安装前必做的几件事

1) 同步 Internet 时间，邮件服务器对时间的要求非常严格，所以这步必须做，如下所示：

```
ntpdate 0.centos.pool.ntp.org
```

2) 更改机器的 hostname 名，我们用 Vim 编辑/etc/sysconfig/network 文件，添加如下内容：

```
hostname=mail.cn7789.com
```

3) 编辑/etc/hosts 文件，改动后的内容如下所示：

```
127.0.0.1 localhost localhost.localdomain mail.cn7788.com mail
```

4) 修改系统的字符编码，如下所示：

```
vim /etc/sysconfig/i18n LANG="en_CN.UTF-8"
```

完成以上步骤后重启服务器，确保在安装前机器的 hostname 名字无误。

5. 安装中牵涉的几个主要用户

安装中牵涉的几个主要用户如下所示：

- ☐ 管理 MySQL 数据库的账号为 root 用户，密码 password。
- ☐ 虚拟域的数据管理员 vmailadmin，密码 password。
- ☐ 虚拟域管理员 postmaster@cn7789.com，密码 password。
- ☐ 超级域管理员 admin@cn7789.com，密码 password。

6. 安装中的注意事项

1) centosplus 的 yum 源相当于是 Centos 官方 yum 的补充，这在当时是一个折中的方案，现在大家可以用 epel 和 rpmforce 的新源来作为补充了。

2) 因为我们的服务器选择的工作方式是图形，所以安装完后 pysieved 必须要在 runlevel5 下也是开启服务的（默认为不开启），开启此服务并设置每次运行均启动，如下所示：

```
service pysieved start
chkconfig pysieved on
```

3) 由于大家都习惯用 http://mail.cn7789.com 方式来访问 Webmail 邮箱，而 iRedmail 原先默认的访问方式为 http://mail.cn7789.com/webmail，所以我们这里可以建立软链接的方式来修正此问题，如下所示：

```
cd /var/www/html
ln -s /var/www/roundcubmail-0.2-stable index.html
```

重启 Apache 服务后就可以用 http://mail.cn7788.com 访问了。

4) 在安装过程中，系统会让你选择 SPF 或 DKIM，二者皆选也可，不过我这里只选择了 SPF，

大多数 DNS 注册商都会支持它，包括新网和万网。

9.4.3 Postfix 本身的防垃圾功能

Postfix 是一个非常优秀的 MTA，它素以高效、安全而著称。Postfix 是我在 Unix 上所见过的 MTA 中在反垃圾邮件（Anti-Spam 或 Anti-UCE）方面做得最好的一个，甚至有很多公司在 Postfix 代码的基础上进行二次开发进而推出反垃圾邮件网关产品。iRedmail 正是在 Postfix 的基础上开发而来的，Postfix 本身的防垃圾功能 iRedmail 都具备了。我们这里可以了解一下相关技术。

MTA 的反垃圾邮件功能，实际上就是在 MTA 处理邮件的过程中对会话进行过滤。这种过滤不但会过滤发往自身的垃圾邮件，而且还会防止自身被恶意利用发送垃圾邮件。Postfix 实现了目前所有主要的 MTA 过滤技术。

MTA 过滤分为两类：邮件数据发送前过滤和邮件数据发送后过滤。本节主要介绍邮件数据发送前过滤。

数据发送前过滤

数据发送前过滤是指在 SMTP 会话中，DATA 指令发送前所进行的过滤。在这个阶段，有 4 种不同子阶段的过滤：SMTP 连接时过滤、HELO/EHLO 指令过滤、MAIL FROM 指令过滤和 RCPT TO 指令过滤。根据这 4 个子阶段接收到信息的不同，它们也分别称做 SMTP 客户端限制、HELO/EHLO 主机名限制、发送者地址限制和接收者地址限制。

过滤默认是在 RCPT TO 指令后生效的，这是因为一些 Windows 上的邮件客户端不处理在 RCPT TO 指令前的过滤动作。可以通过将 smtpd_delay_reject 设置为 no 来使过滤动作立刻生效。这个参数还影响了在不同的指令上可以使用的过滤规则参数。

这 4 个子阶段的过滤分别是通过 4 个配置语句来指定过滤规则的。它们都接收一系列的规则参数列表，参数间可以用空格或逗号分隔开。在默认状态下 smtpd_delay_reject 的值是 yes，它们可以接受所有支持的规则，不过需在 RCPT TO 指令后才全部生效；如果把 smtpd_delay_reject 设置为 no，它们只接受 5 个公共的规则参数、之前子阶段的过滤规则参数和该子阶段的规则参数。

它们接受的公共规则参数如下所示：

permit

表示允许该连接进行。该规则通常置于规则列表的最后面使规则更清晰。

defer

通知客户端现在不能继续会话，稍后再进行 SMTP 连接请求。这常用于服务器需要进行一些 DNS 检查，但是（由于 DNS 查询超时）没有及时获得结果时，它会通知客户端稍候再进行连接。该规则通常置于规则列表的最后面使规则更清晰。

reject

拒绝该连接请求。在这个阶段就断开了连接，有效地节约了垃圾邮件所造成的带宽和处理能力的浪费。该规则通常置于规则列表的最后面使规则更清晰。

拒绝动作默认不会在匹配了拒绝规则后就立刻断开连接，而是在 RCPT TO 指令处理完之后再断开，这是由于一些 Windows 上有缺陷的邮件程序不处理在 RCPT TO 指令前所发回的拒绝状态码。

可以通过将 `smtpd_delay_reject` 设置为 `no` 来立刻发送拒绝状态码断开连接。

`reject_code` 指定了拒绝的返回状态码（默认是 554）。

`warn_if_reject`

改变其后规则的拒绝动作作为警告，即如果其后存在满足拒绝的条件，并不会实际拒绝，而是发出一条警告信息（`reject_warning`）到日志文件中（通常是 `/var/log/maillog`）。它常用于在实际运行的邮件服务器上测试邮件过滤规则。

`reject_unauth_pipelining`

拒绝在 Postfix 支持指令流水线前发送 SMTP 指令流水线的客户端连接。指令流水线是一些邮件客户端为了快速发送邮件所采用的技术。

以下就 4 个子阶段分别来讲述过滤规则。

（1）SMTP 连接时过滤（SMTP 客户端限制）

Postfix 可以在接受客户端的 SMTP 连接请求时进行过滤检查。

通过 Postfix 的 `smtpd_client_restrictions` 指令可以指定这个阶段的过滤规则。这个阶段可用的过滤规则除公共规则外还有如下几条：

`reject_unknown_client`

拒绝客户的地址没有对应 DNS 的 A 记录或 PTR 记录的连接。通常有些机器，尤其是个人拨号用户的机器没有对应的 A 记录或 PTR 记录，所以要注意是否是漫游用户（漫游用户是指不在 `$mynetworks` 中，比如在别的 ISP 拨号上网的用户。通常用 SMTP 认证来解决这个问题）。

`unknown_client_reject_code`

指定了拒绝的返回状态码（默认是 450）。

`permit_mynetworks`

允许来自其 IP 地址属于 `$mynetworks` 所定义网络的客户端连接。通常可用于 ISP 为自己的拨号用户提供 SMTP 服务时，通过 `$mynetworks` 参数指定自己的网络并允许自己网络内的机器发送邮件。

`reject_rbl_client domain.tld`

`reject_rhsbl_client domain.tld`

拒绝对来自 RBL 和 RHSBL 列表中的地址进行连接。通过检查一个 IP 地址或域名是否存在于 `domain.tld` 的 RBL 或 RHSBL 中，可以判断该客户端是否被列入了 `domain.tld` 的实时黑名单，从而决定是否接受连接。

`maps_rbl_reject_code`

指定了拒绝的返回状态码（默认是 554）。

`check_client_access maptype:mapname`

搜索名为 `mapname` 的 `maptype` 类型的访问数据库。可以根据客户端的主机名、父域、IP 地址或部分 IP 地址来匹配。

下面来举一个例子，如下所示：

```
smttd_client_restrictions = hash:/etc/postfix/access,
reject_rbl_client relays.ordb.org,
reject_rhsbl_client dsn.rfc-ignorant.org,
permit_mynetworks,
reject_unknown_client
```

例子中 relays.ordb.org 和 dsn.rfc-ignorant.org 都是国外比较权威的免费 RBL 和 RHSBL 服务器。

(2) HELO/EHLO 指令过滤 (HELO/EHLO 主机名限制)

在接受了 SMTP 连接后, 可以对 HELO 或 EHLO 指令所发送的信息进行过滤检查。有些邮件客户端在通信时并不发送 HELO/EHLO 指令, 可以通过将 smtpd_helo_required 设置为 yes 强制要求发送 HELO/EHLO 指令 (默认 Postfix 不要求发送 HELO/EHLO)。

通过 Postfix 的 smtpd_helo_restrictions 指令可以指定这个阶段的过滤规则。这个阶段可用的过滤规则除公共规则和 smtpd_client_restrictions 的规则外还有如下几条:

```
reject_invalid_hostname
```

拒绝无效格式的主机名连接。

```
invalid_hostname_reject_code
```

指定了拒绝的返回状态码 (默认是 501)。

```
reject_unknown_hostname
```

拒绝未知的主机名连接。所谓未知的主机名是指该主机没有 DNS 的 A 记录或 MX 记录。由于很多拨号用户的机器并没有对应的 A 记录或 MX 记录, 所以要注意是否是漫游用户。

```
unknown_hostname_reject_code
```

指定了拒绝的返回状态码 (默认是 450)。

```
reject_non_fqdn_hostname
```

拒绝主机名不是 FQDN 格式 (完全限定域名格式, 即用点分隔开的包括域名和主机名的主机全名) 的连接。

```
non_fqdn_reject_code
```

指定了拒绝的返回状态码 (默认是 504)。

```
permit_naked_ip_address
```

允许直接使用 IP 地址的连接。通常在 HELO/EHLO 中使用主机名而不是 IP 地址。

```
check_client_access maptype:mapname
```

搜索名为 mapname 的 maptype 类型的访问数据库。可以根据 HELO/EHLO 发送的主机名、父域来匹配。

(3) MAIL FROM 指令过滤 (发送者地址限制)

在接受了 SMTP 连接, 客户端发送了 HELO/EHLO 指令后 (该指令可选), 应该通过 MAIL FROM 指令声明发送者的身份, 可以对发送者身份进行过滤检查。

按照 RFC 规范, 在 MAIL FROM 指令和下面的 RCPT TO 指令中应该使用 RFC 821 格式的邮件地址

(例如: <user@domain.tld>), 但是有许多邮件客户端并不规范, 它们往往没有使用标准的 RFC 821 格式。Postfix 默认接受任何可以理解的邮件地址, 如: 丢失了地址里一对尖括号的邮件地址、可以包含 RFC 822 格式注释的邮件地址等。如果希望打开对 RFC 821 格式的限制, 可以将 `strict_rfc821_envelopes` 设置为 `yes`。

通过 Postfix 的 `smtpd_sender_restrictions` 指令可以指定这个阶段的过滤规则。这个阶段可用的过滤规则除了公共规则、`smtpd_client_restrictions` 的规则和 `smtpd_helo_restrictions` 的规则外还有如下几条:

`reject_unknown_sender_domain`

拒绝发送者邮件的域没有 DNS 的 A 记录或 MX 记录的连接。

`unknown_address_reject_code`

指定了拒绝的返回状态码 (默认是 450)。当进行 DNS 查询出现临时错误时 (如查询超时) 也总是返回 450。

`reject_rhsbl_sender domain.tld`

拒绝发送者邮件的域属于 RHSBL 黑名单的连接。通过检查一个域名是否存在于 `domain.tld` 的 RHSBL 中, 可以判断该客户端是否被列入了 `domain.tld` 的实时黑名单, 从而决定是否接受连接。

`maps_rbl_reject_code`

指定了拒绝的返回状态码 (默认是 554)。

`check_sender_access maptype:mapname`

搜索名为 `mapname` 的 `maptype` 类型的访问数据库。可以根据发送者的邮件地址、名字、域和父域来匹配。关于访问数据库请参阅后面的附录 A。

`reject_non_fqdn_sender`

拒绝发送者邮件的域不是 FQDN 格式的连接。

`non_fqdn_reject_code`

指定了拒绝的返回状态码 (默认是 504)。

`reject_sender_login_mismatch`

拒绝发送者在 `$smtpd_sender_owner_maps` 中的用户名和 SASL 登录名不一致的连接。

(4) RCPT TO 指令过滤 (接收者地址限制)

在 MAIL FROM 指令后要通过 RCPT TO 指令指定邮件接收者。可以对接收者身份进行过滤检查, 通过 Postfix 的 `smtpd_recipient_restrictions` 指令可以指定这个阶段的过滤规则。同以上的检查指令不同的是, 为了避免开放转发, 这个指令有默认值: `permit_mynetworks`、`reject_unauth_destination`。这个阶段可用的过滤规则除了公共规则、`smtpd_client_restrictions` 的规则、`smtpd_helo_restrictions` 的规则和 `smtpd_sender_restrictions` 的规则外还有如下几条:

`permit_auth_destination`

Postfix 默认转发以下的邮件:

来自\$mynetworks 中地址发送的邮件；发往\$relay_domains 中的域或其子域的邮件，但是不能包含邮件路由（如 user@elsewhere@domain.tld）。

Postfix 默认接收最终投递目标符合如下条件的邮件：

- ☐ 目标在\$inet_interfaces
- ☐ 目标在\$mydestinations
- ☐ 目标在\$virtual_alias_domains
- ☐ 目标在\$virtual_mailbox_domains

reject_unauth_destination

拒绝不是发往默认转发和默认接收的连接。

relay_domain_reject_code

指定了拒绝的返回状态码（默认是 554）。

permit_mx_backup

允许接收本地主机是邮件投递目标的 MX 地址的邮件，但是不能包含邮件路由（如 user@elsewhere@domain.tld）。

check_recipient_access maptype:mapname

搜索名为 mapname 的 maptype 类型的数据库。可以根据接收者邮件的邮件地址、名字、域和父域来匹配。

check_recipient_maps

拒绝接收者不匹配如下列表的连接：

```
$local_recipient_maps($mydestinations和$inet_interfaces)
$virtual_alias_maps($virtual_alias_domains)
$virtual_mailbox_maps($virtual_mailbox_domains)
$relay_recipient_maps($relay_domains)
```

空的\$local_recipient_maps 和\$local_recipient_maps 表示不对接收者地址进行过滤检查。

Postfix 默认在接收者检查列表的最后做 check_recipient_maps 检查。

reject_unknown_recipient_domain

拒绝接收者邮件的域没有 DNS 的 A 记录或 MX 记录的连接。

unknown_address_reject_code

指定了拒绝的返回状态码（默认是 450）。当进行 DNS 查询出现临时错误时（如查询超时）也总是返回 450。

reject_rhsbl_recipient domain.tld

拒绝接收者邮件的域属于 RHSBL 黑名单的连接。通过检查一个域名是否存在于 domain.tld 的 RHSBL 中，可以判断该客户端是否被列入了 domain.tld 的实时黑名单，从而决定是否接受连接。

maps_rbl_reject_code

指定了拒绝的返回状态码（默认是 554）。

```
reject_non_fqdn_recipient
```

拒绝接收者邮件的域不是 FQDN 格式的连接。

```
non_fqdn_reject_code
```

指定了拒绝的返回状态码（默认是 504）。

```
permit_sasl_authenticated
```

允许通过了 SASL 认证的用户发送邮件。通过 SASL 协议（包括 SASL1 和 SASL2）实现的 SMTP 认证功能需要在编译 Postfix 时编译进去，并在 main.cf 中将 smtpd_sasl_auth_enable 设置为 yes。有关 SASL 及如何在 Postfix 中实现 SASL 认证请大家参考前面的内容（9.3.2 节）。通过 SASL 认证可以对漫游用户提供发信支持，这是关闭 Open-Relay 的重要手段。

iRedmail 邮件系统的核心是 Postfix，它自带的强大垃圾处理能力也被 iRedmail0.4.0 继承了，这也是我们当初架设邮件服务器时所考虑的，实际上我们的邮件系统上线以后，所收的垃圾邮件确实比之前用的别的邮件系统少得多了，同事和客户都反映不错，达到了我们预期的目标，整个邮件项目搭建成功。

9.4.4 iRedmail0.4.0 特有的防垃圾技术

iRedmail0.4 防垃圾技术的强大之处在于它在 Postfix 本身的防垃圾机制的基础上又使用了 Policyd + SpamAssassin 机制；Policyd 可以实现强大的垃圾过滤功能，而 SpamAssassin 是一个区分垃圾邮件和非垃圾邮件的优秀工具，这两种机制都是 iRedmail 的防垃圾技术，它们结合起来使用效果更好。下面分别向大家介绍一下这两种技术。

首先来看一下 SpamAssassin。

SpamAssassin 是一种安装在邮件伺服主机上的邮件过滤器，用来辨别垃圾邮件。它使用大量的预设规则来检查垃圾信，这些规则会检查寄到网域内所有邮件的标头、内文及送信者。它采取的过滤方式是记分制，也就是说会根据我们所设定的标准来给予分数，当超过标准值的时候即判定为 SPAM（垃圾邮件）。

SpamAssassin 有以下优势：

- SpamAssassin 使用大量不同类型的规则和权重来判断垃圾邮件，那些已经被证实有效的规则会被赋予较高的权重以区分垃圾邮件和非垃圾邮件。
- 既便于调整每一个规则的得分，又便于添加新的、基于正则表达式的判断规则。
- SpamAssassin 适用于各种系统的电子邮件环境，可快速识别受信源或识别新的垃圾邮件类别。
- 通过建立一个俗称为“垃圾邮件陷阱”的电子邮件地址（这种电子邮件地址的唯一作用就是将所有的垃圾邮件都转发给垃圾邮件信息交换中心），SpamAssassin 能够随时向不同的垃圾邮件信息交换中心报告新的垃圾邮件信息。
- SpamAssassin 是一个自由软件，它基于 GUN 的公共许可或是 Perl 的 AL 进行分发。这两个分发许可都允许用户自由地修改该软件，并强制其以相同的分发许可再次分发该修改。

下面介绍一下 SpamAssassin 的工作原理。

SpamAssassin 可以通过以下方法来检验一封电子邮件到底是不是垃圾邮件：

- 检查电子邮件的首部信息是否符合各种 Internet 标准（例如：数据的格式是否正确）。
- 检查电子邮件的首部和内容部分的信息是否包含一些垃圾邮件中常见的、用各种语言写成的短语或句子（例如：“快速致富”或是退订该邮件的一些方法描述）。
- 将首部和内容部分的“校验和”与垃圾邮件信息在线数据库中的多个“校验和”作比较来确认垃圾邮件。
- 检查邮件发送者的 IP 地址是否包含在一些在线的站点列表中（这是一些已经被垃圾邮件发送者利用，或是怀疑被垃圾邮件发送者利用的站点）。
- 自定义地址、主机名和域名的白名单（白名单中的邮件不会作为垃圾邮件）和黑名单（上了黑名单的邮件会被当做垃圾邮件）。SpamAssassin 可以根据用户邮件的历史记录自动构建白名单。
- 使用人为指定的一些垃圾邮件（通常称为 spam）和非垃圾邮件（通常称为 ham）的例子来训练 SpamAssassin 识别用户收到的各种不同的邮件。
- 使用 SPF 协议（Sender Policy Framework）比较邮件发送系统的 IP 地址和邮件发送人的域名，来确定该邮件发送系统是否允许该域名下的用户发送邮件。这是 SpamAssassin 3.0 中增加的新特性（这是强大的新功能）。
- SpamAssassin 会优先排除那些愿意使用 Hashcash 形式的邮件发送者（使用 Hashcash 形式会作一些附加计算，消耗一些时间）。因为如果垃圾邮件发送者支持这些附加计算，就不能迅速发送大量的垃圾邮件了，这是 SpamAssassin 3.0 中增加的新特性。
- SpamAssassin 将信息格式验证、内容过滤和参考网络黑名单这 3 种方式相结合使用。
- 在一个挑战/应答系统中，系统会暂存一个陌生发送者发送的所有信息，并返回一个特定代码或是一组指令作为一个挑战。陌生发送者必须以某种方式作出应答，以验证其邮件地址，并证实（通常使用交谈的方式）发送者是一个人而不是一个自动垃圾邮件发送程序。如果发送者应答成功，系统就会真正接收该信息了。
- 在单系统中，当邮件服务器遇到陌生的邮件发送人或是陌生的邮件发送系统时，它先返回一个临时的 SMTP 失败代码。如果上述的邮件发送系统在经过一个合理的时间间隔后，试图重新发送邮件，邮件服务器会正常接收该邮件，以及随后来自于该邮件发送系统的其他邮件（注：邮件服务器认为这不是一个发送垃圾邮件的邮件发送系统，因此同意接收邮件）。因为当垃圾邮件发送者遇到上面的临时 SMTP 错误时，通常会把它当做永久性的错误，或是试图在灰名单规定的时间间隔内连续多次发送该邮件，这样邮件服务器就会拒绝接收来自这个邮件发送系统的邮件（注：邮件服务器认为这就是一个发送垃圾邮件的邮件发送系统，因此拒绝接收邮件）。

看完了以上内容，也许你正惊叹 SpamAssassin 强大的过滤垃圾邮件功能，或许你也可以了解一下 Policyd 的工作机制，Policyd 的原理其实很简单，它是通过操作 MySQL 数据库来实现屏蔽垃圾邮件的，在操作之前我们记得备份一下数据库。

Policyd 的控制功能如下。

1) 我们可以通过加入黑名单来达到屏蔽网段、主机级用户服务的目的，如下所示：

```
mysql> USE policyd;
```

屏蔽单个主机，如下所示。


```
INSERT INTO blacklist (_blacklist) VALUES '192.168.0.1');
```

表示整个网段，可以用 '%' 表示，如下所示：

```
INSERT INTO blacklist (_blacklist) VALUES ('192.168.0.%');
INSERT INTO blacklist (_blacklist) VALUES ('192.168.%.%');
```

表示将单个用户列入黑名单，如下所示：

```
INSERT INTO blacklist_sender (_blacklist, _description) VALUES ('camis@mweb.co.za', '# blacklist single address');
```

表示将整个域列入黑名单（注意：@domain.ltd 并不会将它的子域名也列入黑名单，例如 @sub.domain.ltd），如下所示：

```
mysql> INSERT INTO blacklist_sender (_blacklist, _description) VALUES ('@mweb.co.za', '# blacklist entire domain');
```

2) Recipient Throttle 收件控制。每次 Postfix 收到一封外部 MTA 发来的邮件时，Policyd 都会在 policyd.throttle_rcpt 表里生成或更新对应的记录，例如：

```
rcpt count_max count_cur date time_limit count_tot abuse_cur abuse_tot log_warn
log_panic
www@example.com 3 3 1216362410 3600 1 1 0 0 0
```

下面说明一下此表对应的各项参数：

- count_max：最多能接收的邮件数量。注意这里的 'count_max' 字段，它是用于限制邮件数量的，它表示这个用户在指定的时间段里，最多能接收多少封邮件。超过限定的邮件数，将以 5xx 的错误代码拒绝对方。
- count_cur：表示当前已经接收了多少封邮件。
- time_limit：表示时间周期，以秒为单位，3600 秒表示大于等于 1 小时。

3) 针对单个用户的收件限制。iRedmail 中使用的 policyd-1.82 加了一个补丁：rcpt_acl.patch。打了此补丁之后，可以在 Policyd 里针对收件的单个域、单个用户（Per-User）做限制。例如，不允许所有 hotmail.com 的用户给本域的 www@a.cn 发邮件，但是允许 www@hotmail.com 这个用户发邮件。可以这样来实现以下功能：

- 开启 RCPT ACL 功能，并重启 Policyd 服务。

```
# Part of file: /etc/policyd.conf
RCPT_ACL=1
```

在 MySQL 表中加入如下限制：

```
mysql> USE policyd;
```

关于 rcpt_acl 表结构，我们可以用命令 desc 查看，如下所示：

```
mysql> desc rcpt_acl;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| _sender    | char(60)      | NO   | PRI |          |       |
```

```

|_rcpt      |char(60)      |NO  |PRI |      |      |
|_wblst     |char(60)      |NO  |    |      |      |
|_priority  |int(10) unsigned|NO  |    |0    |      |
+-----+-----+-----+-----+-----+-----+

```

限制整个域时需要加 '@' 符号。如: '@domain.ltd', 命令如下:

```
INSERT INTO rcpt_acl (_sender, _rcpt, _wblst, _priority) VALUES ('@hotmail.com', 'www@a.cn', 'b', 0);
```

限制允许单个用户, 命令如下:

```
INSERT INTO rcpt_acl (_sender, _rcpt, _wblst,
_priority) VALUES ('www@hotmail.com', 'www@a.cn', 'w',
10);
```

这里的 wblst 字段表示是 whitelist 还是 blacklist, 分别用 w 和 b 来体现。

这里的 priority 字段表示优先级。数字越高, 表示优先级别越高。

4) Spamtrap 是 Policyd 里设计的钓鱼机制。它的设计基于这样一个简单的原理: 如果你从未公布过某个邮件地址, 例如, admin@domain.ltd, 并且你和你的同事也从未告知客户发送邮件到这个地址, 或者 admin@ 这个邮件地址根本就不存在, 但是对方还是尝试将邮件投递到 admin@ 中, 那么对方就有 99.9999% 的可能是垃圾邮件发送者。目前有大量的垃圾邮件发送者都是采用猜测和遍历的方式来将常用的用户名作为收件人的, 例如: admin@、postmaster@、administrator@、hr@、job@、zhaopin@、chengdu@、anhui@、guangzhou@ 等, 然后它会尝试将广告垃圾邮件发送到你的邮箱中。用邮件的朋友应该比较了解, 这种发垃圾邮件的方式还是很普遍的。在 policyd v1 的 spamtrap 机制中, 会将这样的发送人的 IP 地址直接放入黑名单中, 默认要 7 天后这个黑名单才会过期。其实现方法如下所示:

在 Policyd 数据库的 Spamtrap 表中插入这个鱼饵信息, 如下所示:

```
# mysql -uroot -p
mysql> USE policyd;
INSERT INTO spamtrap (_rcpt, _active) VALUES ('admin@example.com', 1);
```

在 Policyd 的配置文件/etc/policyd.conf 里再次确认是否开启了 Spamtrap 功能。

```
# Part of file: /etc/policyd.conf
SPAMTRAPPING=1
```

如果没有启用, 需要将 SPAMTRAPPING 的值设置为 1, 然后重启 Policyd 服务。如果已经启用, 则不需要重启 Policyd。

为了便于调试, 可以在/etc/policyd.conf 中将 Spamtrap 的拒收信息修改如下:

```
# Part of file: /etc/policyd.conf
SPAMTRAP_REJECTION='Spamtrap, go away.'
```

然后在 Postfix 的日志文件/var/log/maillog 中跟踪是否出现类似的信息。

```
Jul 9 16:06:49 mailServer policyd: rcpt=4, spamtrap=new, host=209.85.142.184 (unknown),
from=xxx@gmail.com, to=admin@example.com, size=0, expire=1216195609
```

假如出现了这样的信息, 再检查一下数据库 Policyd 中的 blacklist 表是否多了一条记录。如果确

实增加了，那说明这个功能已经正常工作了。

9.4.5 iRedmail0.4.0 是如何利用 ClamAV 防病毒的

Clam AntiVirus (ClamAV) 是免费而且开放源代码的防毒软件，软件与病毒码的更新皆由社群免费发布。目前 ClamAV 主要用在由 Linux、FreeBSD 等 Unix-like 系统架设的邮件服务器上，以提供电子邮件的病毒扫描服务。ClamAV 本身是在文字接口下运作的，但也有许多图形接口的前端工具可用，另外由于其开放源代码的特性，在 Windows 与 Mac OS X 平台都有其移植版。

其主要特征如下：

- 命令行扫描程序。
- 快速、支持按访问扫描的多线程监控程序。
- 支持 Sendmail 的 Milter 接口。
- 支持脚本更新和数字特征库的高级数据库更新程序。
- 支持病毒扫描程序 C 语言库。
- 支持按访问扫描。
- 每天多次更新病毒库总数。
- 内置了对包含 Zip、RAR、Tar、Gzip、Bzip2、OLE2、Cabinet、CHM、BinHex、SIS 及其格式在内的多种压缩包格式的支持。
- 内置了对绝大多数邮件文件格式的支持。
- 内置了对使用 UPX、FSG、Petite、NsPack、wpack32、MEW、Upack 压缩，以及用 SUE、Yoda Cryptor 和其他程序模糊处理的 ELF 可执行文件和便携式可执行文件的支持。
- 内置了对包括 MS Office 和 MacOffice 文件、HTML、RTF 和 PDF 在内的主流文档格式的支持。

官方网站地址：<http://www.clamav.net>

由于目前阶段如果我们的邮件服务器发送出的邮件带病毒的话也会被加进黑名单，所以防毒和杀毒的工作也不可轻视。freshclam 命令每天会定期升级，但由于网络延迟或升级人数增多等原因，有时病毒库升级会失败，所以我将其写进了 Crontab 计划表，这样就会每天定期升级两次，同时也会对 /home 进行杀毒。添加的计划任务 /etc/crontab 表的内容如下：

```
26 3 * * * root /usr/bin/freshclam --quiet >> /dev/null 2>&1
0 6 * * * root /usr/bin/clamscan -r /home >> /dev/null 2>&1
```

前面是 ClamAV 利用 freshclam 程序自行升级自己的病毒库，由于用户邮件主要集中在 /home 目录中，所以我们每天对此进行扫描即可。另外，我们也可以随意或不定时地对整个硬盘进行扫描。

9.4.6 iRedmail0.4.0 邮件服务器的网络安全

由于此邮件服务器是置于电信 IDC 机房内的，考虑到成本问题，所以没有投入硬件防火墙。因此，我们需要做如下安全措施：

- 1) 服务器配置了 DenyHosts 防暴力破解工具，事实证明这个工具的效果达到了我们的预期目标。
- 2) 服务器配置了 PortSentry 工具防止来自网络的恶意的或探测性的扫描，由于服务器带宽和 CPU 资源有限，我不想将这些宝贵的资源浪费在这些无聊的扫描上面。

3) 开启了邮件服务器的 iptables 防火墙。系统默认的 iptables 脚本如下:

```
#/bin/sh
iptables -F INPUT
iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
# http/https, smtp/smtps, pop3/pop3s, imap/imap, ssh, ftp
iptables -A INPUT -p tcp -m multiport --dport
80,443,25,465,110,995,143,993,587,465,22 -j ACCEPT
```

iptables 防火墙运行后, 我们可以运行 Nmap 工具进行扫描, 命令如下所示:

```
[root@mail postfix]# nmap -P0 -sS 211.147.x.x
```

结果如下所示:

```
Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2009-03-29 16:21 CST
Interesting ports on 211.147.x.x:
Not shown: 1668 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
80/tcp    open  http
110/tcp   open  pop3
111/tcp   open  rpcbind
143/tcp   open  imap
443/tcp   open  https
465/tcp   open  smtps
587/tcp   open  submission
993/tcp   open  imaps
995/tcp   open  pop3s
1014/tcp  open  unknown
```

在这里, 我们又发现一个 1014 端被某个进程打开了, 用 `lsof -i:1014` 查看发现还是 `rpc.statd` 打开的, 此服务每次用的端口都不一样啊。本来想置之不理的, 但是 `rpc.statd` 不能正确处理 `SIGPID` 信号, 远程攻击者可利用这个漏洞关闭进程, 进行拒绝服务攻击。发现 `rpc.statd` 是由服务 `nfslock` 开启的, 我们查询了它是一个可选的进程, 它允许 NFS 客户端在服务器上对文件加锁。这个进程对应于 `nfslock` 服务, 我们关掉此服务即可。

系统刚安装时会开启 `rpc.statd` 服务, 它会随机开放端口 (1014 等), 黑客可利用它不能正确处理 `SIGPID` 信号的漏洞关闭进程, 从而进行拒绝服务攻击, 所以强烈建议关闭之, 命令如下:

```
service nfslock off && chkconfig nfslock off
```

其实由于此机是专业的提供邮件服务器的机器, 所以开启 NFS 服务也没什么必要, 我们完全可以禁止此服务, 用如下命令:

```
service portmap stop && service nfs stop
chkconfig portmap off && chkconfig nfs off
```

后期我自己完善了 iptables 脚本, 主要是加入了防 SYN 攻击及 ping 攻击等的功能, 代码如下:

```
#!/bin/bash
```



```

iptables -F
iptables -F -t nat
iptables -X

iptables -P INPUT DROP
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT

#load connection-tracking modules
modprobe ip_conntrack
modprobe iptable_nat
modprobe ip_conntrack_ftp
modprobe ip_nat_ftp

#iptables -A INPUT -f -m limit --limit 100/sec --limit-burst 100 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type echo-request -m limit --limit 1/s --limit-burst 10
-j ACCEPT
#iptables -A INPUT -p tcp -m tcp --tcp-flags SYN,RST,ACK SYN -m limit --limit 20/sec --limit
-burst 200 -j ACCEPT

iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -p tcp -m multiport --dport 80,443,25,465,110,995,143,993,587,465,22 -j ACCEPT

```

4) 在邮件服务器上线投入使用之后,发现还是有些恶意 IP 疯狂连接 25 端口,后来在北京公司唐老师的指导下,编写并完善了下列安全脚本,具体功能介绍请大家参考第 7 章的相关内容,这里就不再说明了。脚本代码如下:

```

#!/bin/bash
netstat -an | grep :25 | grep -v 127.0.0.1 | awk '{ print $5 }' | sort | awk -F: '{print $1,$4}' |
uniq -c | awk '$1 > 50 {print $1,$2}' > /root/black.txt

for i in `awk '{print $2}' /root/black.txt`
do
COUNT=`grep $i /root/black.txt | awk '{print $1}'`
DEFINE="50"
ZERO="0"
if [ $COUNT -gt $DEFINE ];
then
grep $i /root/white.txt > /dev/null
if [ $? -gt $ZERO ];
then
echo "$COUNT $i"
iptables -I INPUT -p tcp -s $i -j DROP
fi
fi
done

```

5) 考虑到后期还要安装 Wireshark 和 NTOP 等图形化监控工具,所以我在此服务器上安装了图

形界面，用如下命令可以找出所有当前运行在 5 级别的服务：

```
chkconfig --list | awk '{print $1 "\t" $7}' | grep 5:on
```

此脚本的作用相当于 ntsysv，不过这个更加直观，Bluetooth、cups、iptables、kudzu、setrouble-shoot 这些服务均可关闭。

另外我在此 iRedmail 邮件服务器上也安装了 tcpdump、iptraf 等监控工具，这些工具也可帮助分析服务器的一些安全问题。

9.4.7 iRedmail0.4.0 邮件服务器系统的监控

关于 iRedmail0.4.0 邮件服务器系统的监控其实也没什么特别的，大家就把 iRedmail0.4.0 邮件服务器当成是一台普通的 Linux 机器监控即可。平时多注意一下 CPU、内存及硬盘还有系统的负载状况，相对而言压力还是来自数据库，因为 iRedmail0.4.0 是基于 LAMP 架构的，前期我们可以将 LAMP 环境部署在一台机器上，后期如果 MySQL 的压力过大，还是要将 MySQL 数据库单独分离到性能比较好的机器上，比如 DELL R710。大家平时可以用 iostat 监控一下磁盘的 I/O。关于邮件的监控我主要做了如下工作：

1) 在 /root/count.sh 下放了查看连接数 enstablishd 的脚本，防止别人恶意攻击时调用 deny_100.sh 来封锁此恶意 IP，内容如下：

```
#!/bin/bash
netstat -an | grep :25 | grep -v 127.0.0.1 | awk '{ print $5 }' | sort | awk -F: '{print $1}' | uniq -c | awk '$1 > 100'
```

因为有手动监控，所以此阈值我定义为 100。

2) 统计服务器连接状态的脚本 /root/summary.sh，内容如下：

```
netstat -n | awk '/^tcp/ { ++S[$NF] } END { for (a in S) print a, S[a] }'
```

执行后的结果如下（下列结果是正常的）：

```
TIME_WAIT 1
ESTABLISHED 268
SYN_RECV 1
CLOSING 1
LAST_ACK 1
```

3) 我的工作习惯是在内网环境下部署 Nagios 进行即时监控，由于此机房内只有一台机器，所以部署 Nagios 机显然有点不划算，所以我写了以 vmstat 为基础的监控脚本（如 /root/monitor.sh），内容如下：

```
#!/bin/bash
while :
do
    vmr='vmstat | tail -1 | awk '{print $1}''
    if [ ${vmr} -gt 4 ]
    then
        date >> /root/monitor.txt
        vmstat >> /root/monitor.txt
    fi
done
```

```

netstat -anp >> /root/monitor.txt
ps -aux >> /root/monitor.txt
last >> /root/monitor.txt
tail -10 /var/log/messages >> /root/monitor.txt
fi
sleep 60
done

```

此脚本可放至后台运行 `nohup sh /root/monitor.sh &`，如果遇到 CPU 繁忙的情况，它会自动记载系统日志等以供分析。

`vmstat` 是 Virtual Memory Statistics（虚拟内存统计）的缩写，可对操作系统的虚拟内存、进程、CPU 活动进行监视。它可对系统的整体情况进行统计，不足之处是无法对某个进程进行深入的分析。

`vmstat` 的语法如下：

```
vmstat [-V] [-n] [delay [count]]
```

其中，`-V` 表示打印出版本信息；`-n` 表示在周期性循环输出时，输出的头部信息仅显示一次；`delay` 是两次输出之间的延迟时间；`count` 是指按照这个时间间隔统计的次数。对于 `vmstat` 输出各字段的含义，可运行 `man vmstat` 查看。

`vmstat` 命令有 4 个可选标志可供使用。如果机器有虚拟地址缓存，`-c` 标志就改变输出报告缓存刷新统计数据。报告包括自从系统启动后每种缓存刷新的全部总量。6 个缓存类型是用户、上下文、区域、段、页、部分页。我在第 1 章也介绍过 `vmstat`，大家可以参考其中 `vmstat` 的详细语法，这里就不再重复了。

我们重点需要关注的内容如下所示：

- 如果 `r` 经常大于 4，且 `id` 数经常小于 40，表示 CPU 的负荷很重。
- 如果 `pi`、`po` 长期不等于 0，表示内存不足。
- 如果 `disk` 经常不等于 0，且在 `b` 中的队列大于 3，表示磁盘 I/O 性能不好。

最重要的是，我们通过 `vmstat` 的结果可以判断出到底是系统中的哪块出现了性能瓶颈。

- 通过 `vmstat` 结果判断 CPU 瓶颈。

`r`（运行队列）展示了正在执行和等待 CPU 资源的任务个数。当这个值超过了 CPU 数目时，就会出现 CPU 瓶颈了。

获得 CPU 个数的命令如下所示：

```
cat /proc/cpuinfo | grep processor | wc -l
```

解决办法大体有如下几种：

- 最简单的就是增加 CPU 个数。
- 通过调整任务执行时间（如大任务放到系统不繁忙的情况下执行），进而平衡系统任务。
- 调整已有任务的优先级。
- 通过 `vmstat` 识别 CPU 满负荷。

首先需要声明的一点是，`vmstat` 中 CPU 的度量值是百分比的。当 `us + sy` 的值接近 100 时，表示 CPU 接近满负荷工作。但要注意的是，CPU 满负荷工作并不能说明什么问题，Unix 总是试图要

CPU 尽可能地繁忙，以使任务的吞吐量最大化。唯一能够确定 CPU 瓶颈的还是 r（运行队列）的值。

□ 通过 vmstat 识别 RAM 瓶颈。

数据库服务器都只有有限的内存，出现内存争用现象是 Oracle 的常见问题。

首先查看 RAM 的数量，命令如下（Centos5.5 环境）：

```
free
      total        used        free      shared    buffers     cached
Mem:   1027348      873312      154036      185736      187496      293964
-/+ buffers/cache:      391852      635496
Swap:   2096440           0      2096440
```

当然也可以使用 top 等其他命令来显示 RAM。

当内存的需求大于 RAM 的数量时，服务器启动虚拟内存机制。通过虚拟内存，可以将 RAM 段移到 SWAP DISK 的特殊磁盘段上，这样会出现虚拟内存的页导出和页导入现象，页导出并不能说明是 RAM 瓶颈，虚拟内存系统经常会对内存段进行页导出，但页导入操作就表明服务器需要更多的内存了，页导入要从 SWAP DISK 上将内存段复制回 RAM 中，这会导致服务器的速度变慢。建议大家多关注一下 swap 信息，如果长期出现 swap used 的情况，我们就应该要查看内存够不够用了。

解决的办法有如下几种：

- 最简单的是加大 RAM。
- 改小 SGA，使得对 RAM 需求减少。
- 减少 RAM 的需求（如：减少 PGA）。

作为邮件系统，网络流量也是一个不容忽视的监控内容。不论是总体吞吐量还是网络 iops，都要给予一定的关注。网络流量的监控我们可以通过 iptraf 工具或自行编辑 SHELL 脚本来实现，还可以利用 NTOP 或 MTRG 这些工具来完成，或者对网络交换机进行监控也可以。这些方法各有各的特点，大家可以根据自身的机房条件来选择如何监控自己的 iRedmail 邮件服务器。

关于日志监控和分析，由于 iRedmail 本身就装了 Awstats 工具，而且此工具做得非常好，大家可以用其来分析哪些 IP 和网站频繁地访问我们的邮件，了解邮件的错误码等信息。

iRedmail 邮件系统的 MySQL 优化方法请大家参考第 7 章中的内容，这里就不再重复了。

9.4.8 iRedmail0.4.0 的系统文件备份

iRedmail0.4.0 是通过 SHELL 脚本备份的，系统自身就带了备份脚本，名字为 backup_iRedMail.sh，官方推荐用此脚本来备份我们的 iRedmail 系统，内容如下：

```
[root@mail ~]# cat backup_iRedMail.sh
#!/bin/sh

# Filename: backup_iRedMail.sh
# Author:  Zhang Huangbin (michaelbibby@gmail.com)
# Purpose: Backup all mail server related software configure files.
# Project: Open Source Mail Server Solution for Red Hat Enterprise
#          Linux and CentOS 5.x:
#          http://code.google.com/p/iredmail/
```



```

# -----
# ----- USAGE -----
# -----
# Run this script as root user:
#
# # sh backup_iRedMail.sh
#
# It will collect mail server related configuration files and other
# stuffs and compress them as a tarball under /root/ by default.
# -----

# Last update: 2008.10.23
# ChangeLog:
# - 08.10.27: Add directory: /var/lib/dovecot/. It maybe contains bdb
#             database of plugin expire.

export BACKUP_DIR='/usr/iRedMail_Backup'
export DATE="$(/bin/date +%Y.%m.%d_%H.%M.%S)"
export BACKUP_TARBALL="${BACKUP_DIR}/iRedMail_Backup - ${DATE}.tar"
export COMPRESSED='YES'
export COMPRESS_CMD='bzip2 -9'

check_user()
{
    # Check special user privilege to execute this script.
    if [ X"${id -u}" != X"${id -u ${1}}" ]; then
        ECHO_INFO "Please run this script as user: ${1}."
        exit 255
    else
        :
    fi
}

# Function for file collection.
collectit()
{
    # $1 -> Tar file or tape device
    # $2, $3, $4, ... -> File(s) to append

    if [ "$#" -ge 2 ]; then
        tarfile="$1"
        shift 1
        tar rfp ${tarfile} $@ >/dev/null
    else [ "$#" -lt 2 ]
        echo "Usage: collectit <tar file or tape device> <file_to_append>"
    fi
}

# Which files we should collect.
FIREWALL='/etc/selinux/config'

```

```
/etc/sysconfig/iptables
```

```
,
```

```
HTTPD="/etc/httpd/  
/etc/logrotate.d/httpd  
/etc/rc.d/init.d/httpd  
/etc/sysconfig/httpd  
"
```

```
PHP="/etc/php.ini"
```

```
OPENLDAP="/etc/openldap/  
/etc/rc.d/init.d/ldap*  
"
```

```
MYSQL="/etc/my.cnf*  
/etc/rc.d/init.d/mysqld  
"
```

```
POSTFIX="/etc/postfix/  
/etc/rc.d/init.d/postfix  
/etc/pam.d/smtp.postfix  
/usr/lib/sasl2/smtpd.conf*  
/usr/lib64/sasl2/smtpd.conf*  
"
```

```
DOVECOT="/etc/dovecot* .conf*  
/etc/logrotate.d/dovecot  
/etc/rc.d/init.d/dovecot  
/home/vmail/.dovecot.sieve  
/var/lib/dovecot/  
/var/www/sieve/  
"
```

```
CLAMAV="/etc/clamd.conf  
/etc/freshclam.conf  
/etc/logrotate.d/clamav  
/etc/logrotate.d/freshclam  
/etc/cron.daily/freshclam  
/etc/rc.d/init.d/clamd  
/etc/rc.d/init.d/freshclam  
"
```

```
AMAVISD="/etc/amavisd.conf  
/etc/cron.daily/amavisd  
/etc/logrotate.d/amavisd  
/etc/sysconfig/amavisd  
/etc/rc.d/init.d/amavisd  
/var/lib/dkim/  
/var/amavis/
```

```

/var/virusmails/
'

SPAMASSASSIN="/etc/mail/spamassassin/
"

PYSIEVED="/etc/pysieved.ini*
/etc/xinetd.d/pysieved
"

MISC="/etc/syslog.conf
/etc/sysctl.conf
/etc/sysconfig/network
/etc/sysconfig/network-scripts/ifcfg-eth*
/etc/sysconfig/saslauthd
/etc/pki/iRedMail*
"

# Do not modify it unless you add new categories.
ALL_FILES="${FIREWALL} ${HTTPD} ${PHP} ${OPENLDAP} ${MYSQL} ${POSTFIX}
${DOVECOT} ${CLAMAV} ${AMAVISD} ${SPAMASSASSIN} ${PYSIEVED} ${MISC}"

# Check user.
check_user root

# Create BACKUP_DIR.
[ -d ${BACKUP_DIR} ] || mkdir -p ${BACKUP_DIR}

# Copy all files we need.
echo -n " * Backup files and directories..."
collectit ${BACKUP_TARBALL} ${ALL_FILES}
echo -e "\tDone."

# Compress.
if [ X"${COMPRESSED}" == X"YES" ]; then
    echo -n " * Compress tarball: ${BACKUP_TARBALL}"
    ${COMPRESS_CMD} ${BACKUP_TARBALL}
    echo -e "\tDone."
else
    :
fi

cat <<EOF

*****
***** WARNING *****
*****
Script does *NOT* backup below files/data, please do it *MANUALLY* .

- MySQL databases (Default: /var/lib/mysql).

```

```
- Users' mailboxes (Default: /home/vmail).
- Postfix queues (Default: /var/spool/postfix).
```

```
EOF
```

9.4.9 iRedmail0.4.0 的 MySQL 数据库备份方案

关于 iRedmail0.4.0 的 MySQL 数据库备份，我前期采用的是 `mysqldump` + `mysqlhotcopy` 双备份的方式。`mysqlhotcopy` 是一个 PERL 程序，最初由 Tim Bunce 编写。它主要的实现原理就是先锁住表 (LOCK TABLES)，然后执行 FLUSH TABLES 动作，该正常关闭的表正常关闭，该进行 flush 同步的数据都进行同步，然后通过执行系统级别的复制命令，将要备份的表或数据库的所有物理文件都复制到指定的备份位置。它是备份数据库或单个表的最快途径，但它只能运行在数据库文件（包括数据表定义文件、数据文件、索引文件）所在的机器上。

这里首先来介绍 MySQL 的 `mysqldump` 备份脚本，`mysqldump` 的脚本我放在了 `/root/backup_mysql_db.sh` 中，`mysqldump` 是 iRedmail 官方推荐的备份方案之一，脚本内容如下：

```
# Author:  Zhang Huangbin (michaelbibby <at> gmail.com)
# Date:    2007.09.16
# Purpose: Backup specified mysql databases.

# Copyright:
#
# This shell script is shipped within iRedMail project, released under
# GPL v2.
# ----

# Usage:
# * Add crontab job for whichever user, such as root:
#
#     # crontab -e -u root
#     1 4 * * * /bin/sh /path/to/backup_mysql_db.sh
#
# * Make sure 'crond' service is running when system startup:
#
#     # chkconfig --level 345 crond on
# ----

# -----
# ---- Modify below variables to suit your need ----
# -----

# Where to store backup copies.
BACKUP_ROOTDIR = '/backup/mysql/'

# MySQL user and password.
MYSQL_USER = 'root'
MYSQL_PASSWD = 'password'

# Which database(s) we should backup.
```



```

DATABASES="mysql vmail policyd roundcubemail"

# Database character set.
DB_CHARACTER_SET="utf8"

# Compress: YES | NO.
COMPRESS="YES"

# Delete plain SQL file after compressed. Compressed copy will be
# remained.
DELETE_PLAIN_SQL_FILE="YES"

# -----
# ---- You do * NOT* need to modify below lines. ----
# -----

MONTH="$(/bin/date +%Y.%m)"
DATE="$(/bin/date +%Y.%m.%d)"

BACKUP_DIR="${BACKUP_ROOTDIR}/db/${DATE}"

# Check necessary directory.
[ -d ${BACKUP_DIR} ] || mkdir -p ${BACKUP_DIR}

# Logfile directory.
LOG_DIR="${BACKUP_ROOTDIR}/logs/${MONTH}/"
[ -d ${LOG_DIR} ] || mkdir -p ${LOG_DIR}

LOGFILE="${LOG_DIR}/backup-mysql-${DATE}.log"

# create and init log file.
echo -e "\nLog init at: ${DATE}." > ${LOGFILE}
echo "Operator: Michael Bibby(michaelbibby@gmail.com)." >> ${LOGFILE}
# ---- End log ----

backup_db()
{
    # USAGE:
    # backup dbname
    output_sql="${BACKUP_DIR}/${1} - ${DATE}.sql"

    mysqldump \
        -u ${MYSQL_USER} \
        -p ${MYSQL_PASSWD} \
        --default-character-set=${DB_CHARACTER_SET} \
        $1 > ${output_sql}
}

# Backup.
for i in ${DATABASES}
do

```

```

        echo -n "Backup database: $i..." >> ${LOGFILE}
        backup_db $i
        echo -e "\tDone" >> ${LOGFILE}
done

# Compress plain SQL file.
if [ X"${COMPRESS}" == X"YES" ]; then
    for i in $(ls ${BACKUP_DIR}/*)
    do
        echo -n "Compress: $i..." >> ${LOGFILE}
        bzip2 $i
        echo -e "\tDone" >> ${LOGFILE}
    done
else
    :
fi

# Size of the backup file.
du -sh ${BACKUP_DIR}/* >> ${LOGFILE}

# Delete plain SQL file after compressed.
if [ X"${DELETE_PLAIN_SQL_FILE}" == X"YES" ]; then
    rm -f ${BACKUP_DIR}/*.sql
else
    :
fi

```

接下来再说说 mysqlhotcopy，我将其放在 /root 下，脚本名字为 backup_mysqlhotcopy.sh，代码内容如下：

```

#!/bin/bash
PATH=/usr/local/sbin:/usr/bin:/bin

# The Directory of Backup
BACKDIR=/usr/mysql_backup
# The Password of MySQL
ROOTPASS=password
# Remake the Directory of Backup
rm -rf $BACKDIR
mkdir -p $BACKDIR
# Get the Name of Database
DBLIST='ls -p /var/lib/mysql | grep / | tr -d /'

# Backup with Database
for dbname in $DBLIST
do
mysqlhotcopy $dbname -u root -p $ROOTPASS $BACKDIR | logger -t mysqlhotcopy
done

```

分别将以上自动备份作业添加至 /etc/crontab 表里，然后在最后添加以下内容：

```
1 12 * * * root /bin/sh /root/backup_mysqlhotcopy.sh >> /dev/null 2>&1
0 2 * * * root /bin/sh /root/backup_iRedMail.sh >> /dev/null 2>&1
```

运行 mysqlhotcopy 出现错误的解决方法如下所示。

错误一：我们有时用 mysqlhotcopy 备份 MySQL 数据库时会报 Invalid db.table name 类似的字句，MySQL 官网推荐用下面的方法解决。

错误描述如下：

```
[root@mail ~]# ./backup_mysqlhotcopy.sh
Invalid db.table name 'mysql.mysql'. 'columns_priv' at /usr/bin/mysqlhotcopy line 855.
Invalid db.table name 'policyd.policyd'. 'blacklist' at /usr/bin/mysqlhotcopy line 855.
Invalid db.table name 'roundcubemail.roundcubemail'. 'cache' at /usr/bin/mysqlhotcopy line 855.
Invalid db.table name 'vmail.vmail'. 'admin' at /usr/bin/mysqlhotcopy line 855.
[root@test ~]# vi /usr/bin/mysqlhotcopy
```

用/dbh_ tables 查找到如下内容：

```
my @dbh_tables = eval { $dbh->tables() };
```

在它下面添加如下一行：

```
map { s/^. * ? \.//o } @dbh_tables;
```

错误二：mysqlhotcopy 由于没有配置 LANGUAGE 等变量执行产生错误。其实大家可以观察如下报错信息，在第三行和第四行就已经清楚说明了：

```
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
LANGUAGE = (unset),
LC_ALL = (unset),
LANG = "en_CN.UTF-8"
are supported and installed on your system.
perl: warning: Falling back to the standard locale ("C").
Invalid db.table name 'mysql.mysql'. 'columns_priv' at
/usr/bin/mysqlhotcopy line 855.
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
LANGUAGE = (unset),
LC_ALL = (unset),
LANG = "en_CN.UTF-8"
are supported and installed on your system.
perl: warning: Falling back to the standard locale ("C").
Invalid db.table name 'policyd.policyd'. 'blacklist' at
/usr/bin/mysqlhotcopy line 855.
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
LANGUAGE = (unset),
LC_ALL = (unset),
LANG = "en_CN.UTF-8"
are supported and installed on your system.
perl: warning: Falling back to the standard locale ("C").
Invalid db.table name
```

```
'roundcubemail.roundcubemail','cache' at
/usr/bin/mysqlhotcopy line 855.
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
LANGUAGE = (unset),
LC_ALL = (unset),
LANG = "en_CN.UTF-8"
    are supported and installed on your system.
perl: warning: Falling back to the standard locale ("C").
Invalid db.table name 'vmail.vmail'.'admin' at
/usr/bin/mysqlhotcopy line 855.
```

具体解决方法如下，大家可以按照如下步骤操作。

我们首先查看系统预设的 LANG 等变量情况，命令如下：

```
echo $LANG
```

显示结果如下所示：

```
en_US.UTF-8:en_US:en_US.ISO-8859-1
```

再查看下 SUPPORTED 变量的情况，命令如下：

```
echo $SUPPORTED zh_HK.UTF
```

命令显示结果如下：

```
8:zh_HK:zh:zh_CN.GB18030:zh_CN:zh:zh_TW.Big5:zh_TW:
zh:en_US.UTF-8:en_US:en:en_US.ISO-8859-1
```

我们继续查看 LANGUAGE 和 LC_ALL 变量时，发现它们均显示为空，即没有设定，命令如下所示：

```
echo $LANGUAGE
echo $LC_ALL
```

我们将其设定下，用 export 命令导出即可以解决 mysqlhotcopy 的这个问题了，如下所示：

```
export LC_ALL=en_US
export LANGUAGE=en_US
```

MySQL 数据库正式上线之后，我打开了 binlog 日志，采用的备份方案是 mysqldump 日备份，mysqlhotcopy 周备份，运行一段时间后，再检查磁盘空间的情况，发现放数据库的分区磁盘激增了 140 多个 GB，查找原因，发现仅 binlog 日志就有 100 多 GB，根源就在这里。再查看 my.cnf，看到 binlog 的 size 是 1GB 就做分割，但没有看到删除的配置。在 MySQL 里显示了一下 variables，命令如下：

```
mysql> show variables like '%log%';
```

查到此参数的相关设定：

```
| expire_logs_days | 0 |
```

这是一个 global 的参数，默认是 0，也就是 logs 不过期，我们可以将其设定为 30 天为一个轮询，命令如下所示：


```
set global expire_logs_days=30;
```

这样 30 天前的 log 就会被删除了，如果有回复的需要，请大家做好备份工作。但这样设置还不行，下次重启 MySQL 时，又会恢复默认配置了，所以我们需要在 my.cnf 中设置：

```
expire_logs_days = 30
```

这样重启也就不受影响了。

想要恢复数据库以前的资料，执行 `mysql > show binlog events;`。

由于日志数据量很多，查看起来很麻烦，光打开个文件就要半天，并且如果使用的时间足够长的话，会把我的硬盘空间都给消耗掉，所以应该适当删除部分可不用的日志。操作步骤如下所示：

1) 登录系统。

```
/usr/bin/mysql
```

使用 MySQL 查看日志。

```
mysql> show binary logs;
+-----+-----+
| Log_name          | File_size |
+-----+-----+
| ablelee.000001    | 150462942 |
| ablelee.000002    | 120332942 |
| ablelee.000003    | 141462942 |
+-----+-----+
```

2) 删除 bin-log (删除 ablelee.000003 之前的但不包含 ablelee.000003)，如下所示：

```
mysql> purge binary logs to 'ablelee.000003';
Query OK, 0 rows affected (0.16 sec)
```

3) 查询结果 (现在只有一条记录了) 如下：

```
mysql> show binlog events \G
***** 1. row *****
Log_name: ablelee.000003
Pos: 4
Event_type: Format_desc
Server_id: 1
End_log_pos: 106
Info: Server ver: 5.1.26 - rc - log, Binlog ver: 4
1 row in set (0.01 sec)
(ablelee.000001 和 ablelee.000002 已被删除)
mysql> show binary logs;
+-----+-----+
| Log_name          | File_size |
+-----+-----+
| ablelee.000003    | 106 |
+-----+-----+
1 row in set (0.00 sec)
```

在 MySQL 数据库的日常维护工作中，我们仅仅做备份是不够的，还应该多吃一些故障模拟实

难和数据恢复实验，以保证数据的完整性和可靠性。

9.4.10 维护 iRedmail0.4.0 邮件服务器的一些注意事项

iRedmail0.4.0 正式上线后，我们平时也要注意多检查和维护，经常关注一下邮件日志和 Awstats 统计信息。我将在维护时遇到的一些情况进行了归纳如下。

1) 有关 Postfix 性能方面的考虑 (iRedmail 作者原话): Postfix 能实现日收发 10KB ~100KB 的信件，在性能方面须注意的是 SpamAssassin 可能会是个性能瓶颈。所以建议刚开始的几天、几个星期都密切关注 maillog，看看反垃圾效果。如果还不错，建议让 Amavisd 不调用 SpamAssassin 做邮件内容扫描了。另外，为了通用，iRedmail 默认在 Postfix 里开启了一些 MySQL 查询，建议你根据自己的需要关闭一些查询，我们可以用如下查看 Postfix 相关的 MySQL 查询，如下所示：

```
postfix -n |grep mysql
```

2) 关于 iRedmail 邮件服务器的时间问题。因为没有上下级的 NTP 时间服务器，所以我关闭了 ntpd 服务，仅仅把本地作为了一个客户端，对时则采用的是 Crontab 方式，即 14 04 * * * root /usr/sbin/ntpdate 210.72.145.44。而 iRedmail 日志采用的是 UTC 方式，跟系统时间 cst 刚好相差 8 小时，如果改动 iRedmail 邮件服务器时间的话，会导致 Dovecot 死机 (Dovecot 会“自杀”)。后来我在 iRedmail 的官方论坛上发帖咨询了此事，发现其他网友也有此反映，有网友建议的解决方案是安装系统时不选择 UTC，这样邮件时间就准确了，但会导致系统时间不准确。由于这不影响整体使用，只是邮件日志时间有所偏差而已，所以我没做任何改动。

3) 定期执行 df -h 命令，检查和整理一下磁盘，因为有的 SHELL 备份脚本，比如备份邮件系统和 mysqldump 的脚本，做的都是全备。另外，我也开启了 MySQL 的 binlog 日志，熟悉 MySQL 的朋友应该知道这一项功能占用磁盘的速度是很惊人的。所以后期可能会发生磁盘空间不足的问题，我们可以手动操作删除以前不必要的备份，以免浪费空间。不太建议写轮询 SHELL 脚本，因为 MySQL 的数据比较重要，万一出现误删除现象会很麻烦的。

4) 在工作是我们遇到了个问题即通过虚拟别名 tech@cn7789.com 向 yuhongchun@cn7789.com、st02@cn7789.com、st02@cn7789.com 发送邮件，但通过 aliases 表实现不了，我在《Postfix 权威指南》上找到了答案，即我们可以通过后台修改或修改 MySQL 数据库的方式来实现。

Alias_map = mysql:/etc/postfix/mysql-aliases.cf 所包含的不是直接可查询的数据，而是描述如何查询 MySQL 数据库的配置信息。

Local_recipient_maps = mysql:/etc/postfix/mysql-local.cf 是所有合法的本地邮件账户的记录。

aliases 是实现本地用户的转发权限，cn7789.com 则是一个虚拟域，其用户管理是由 MySQL 来实现的，这是概念性的东西。

5) 在用 postmaster@cn7789.com 登录 postfixadmin 时，本来想新增一个虚拟域，结果发现老是权限不够。后来通过查询数据库，了解到 MySQL 的 admin_domain 表里的 domain 项中哪个用户是 ALL 权限，一下子就明白问题的所在了。建议大家多了解一下 iRedmail 中 MySQL 各个表所代表的含义，后期的 iRedmail 版本好像有相应的工具推出，不过是收费的。

6) 在 iRedmail 邮件日志中常会遇到 SMTP 错误代码 (以下截图取自于系统自带的 Awstats)，如图 9-5 所示。

□ 550: 所要求的动作无法执行，邮箱不存在，不再尝试投递。

SMTP错误代码			
SMTP错误代码	邮件	百分比	大小
550 Requested mail action not taken: relaying not allowed, unknown recipient user, ...	1243463	84.3 %	0
450 Requested mail action not taken: mailbox busy, DNS check failed or access denied for other reason	230410	15.6 %	0
999 Unknown error	821	0 %	851.29 K字节
554 Requested mail action rejected: access denied	42	0 %	0
504 Command parameter not implemented	42	0 %	0
553 Requested mail action not taken: mailbox name not allowed	18	0 %	0

图 9-5 Awstats 中统计邮件日志中的 SMTP 错误代码

- 450: 所要求的邮件动作无法执行, 邮箱不可用, 邮件保存在本地, 可能会尝试重新投递, 通常这情况发生在每次投递时遇到的意外错误。
- 554: 拒绝不是发往默认转发和默认接收的连接。
- 504: 拒绝接收者的邮件域不是 FQDN 格式的连接。
- 553: 所要求的动作未执行, 不接受该邮箱, 通常发生在邮件地址错误时, 此邮件会被视为垃圾邮件拒收, 不再尝试投递。

7) 分析日志时可结合以上状态码进行分析, 比如, 我想统计一下邮件日志中最后 50 000 行中含有 554 错误代码的日志, 可以用如下方法:

```
tail -n 50000 /var/log/maillog | grep 554
Mar 30 00:54:06 mail postfix/smtpd[10763]: NOQUEUE: reject: RCPT from unknown[122.116.54.156]:
554 5.7.1 <service@c3.counter.idv.tw>: Relay access denied; from=<service@c3.counter.idv.tw>
to=<service@c3.counter.idv.tw> proto=ESMTP helo=<user-yv38ncy07o>
```

参考资料如下:

《鸟哥的 Linux 私房菜: 服务器架设篇》(第 3 版) 人民邮电出版社

《Postfix 权威指南中文版》, Kyle D.Dent, O'Reilly Taiwan 公司编译

iRedmail 官方地址: <http://www.iRedmail.org>, iRedmail

Postfix 官方论坛: <http://www.extmail.org/forum/>

邮件系统的更多注意事项和细节问题需要大家在实际工作中查找挖掘, 并找出解决方案。邮件的维护工作是件很繁琐的工作, 需要大家的耐心和细心, 这在以后的工作中会慢慢体会到。

9.5 小结

通过这章的学习, 大家应该可以初步了解和掌握邮件的搭建工作和基本原理。本章介绍了 Sendmail、Postfix 和 iRedmail 的安装及搭建过程, 在此之前还详细讲解了 DNS 的安装工作。没接触邮件相关概念的朋友可能会觉得无从下手, 如果是这样, 建议你至少掌握 DNS 的原理和安装过程 (yum 或源码任选一种即可)。另外可以掌握 Postfix + Dovecot 邮件系统的搭建过程, 如果有条件的话也可以了解和熟悉一下 Exchange2007 邮件系统, 它结合 Windows Server 系列的域控性能也是非常强大的, 我认识的不少朋友都是专门在企业中做 Exchange 2007 相关工作的。在正式的工作环境中, 由于邮件系统是一个非常专业的工作, 建议大家还是以高度集成的 iRedmail 或类似的系统来搭建邮件系统。如果公司要做专业的邮件系统的话, 那就要花非常多的精力和时间在上面了; 如果公司不是做非邮件系统业务的, 邮件只需满足一般工作需要即可, 建议大家不要花过多的时间在此之上, 3~6 个月足够了。大家应该把精力更多地集中在 Linux 集群和虚拟化、网站架构、安全方面和软件开发等其他方面。



第 10 章

系统管理员在企业中的职业定位及发展方向

- 10.1 系统管理员的概念和工作职责
- 10.2 系统管理员应该熟悉的系统
- 10.3 系统管理员应该熟悉的工具
- 10.4 Linux 的学习及进阶之路
- 10.5 系统管理员应该如何工作
- 10.6 给 Linux/Unix 新人的建议
- 10.7 系统管理员之企业生存守则
- 10.8 小结

10.1 系统管理员的概念和工作职责

大家在看前面的内容时，已经接触到了系统管理员这个概念，可能会有许多朋友对这个职业很感兴趣：这是个怎样的职业呢？主要是做什么工作的？下面我结合工作中的一些体会跟大家介绍一下系统管理员的工作职责。

系统管理员（System Administrator，简称为 System Admin）主要分为网络管理员（也称为网络工程师，即 Network Engineer，简称为网工或网管，因为他们有时还要负责公司员工的计算机和电话系统维护等工作）和系统管理员。网络管理员主要负责整个公司网络的设计，网络设备的安装、配置、管理和维护工作，为内部网的安全运行做技术保障。而公司的服务器是网络应用系统的核心，由系统管理员专门负责管理。

系统管理员的主要工作职责如下：

- 确保公司的业务系统或网站正常运营。这相当不容易，相当于 365 × 24 小时都处于待命状态，出了紧急事件要迅速处理（特别是电子商务网站）。
- 系统管理员对系统账号及密码要认真管理，严禁泄露。若由于管理不当造成数据丢失和破坏，则要承担损失责任。
- 忠于职守，能及时处理工作中的问题，确保公司办公网络正常（有时这块工作由公司网工或网管负责）。
- 加强管理，对于系统出现的故障和存在的隐患，及时向有关部门报告，以便尽快得到处理。
- 负责调换录像带，查看监控录像（有时这块工作由公司网工或网管负责），要公正无私。
- 遵守保密原则，严禁无关人员进入机房和查询统计数据。
- 完成公司上级领导交代的各项工作。

以上是一般企业中系统管理员的工作职责，当然了，根据公司性质不同，系统管理员的工作内容也会随之发生变化。比如我最近所从事的两份工作，虽然都是做系统管理员的工作，但偏重于网站维护和项目实施那部分，交换机和路由交换则接触得较少。

我前一份工作是在一家软件外包公司做系统管理员，其工作职责有以下 3 点：

- 帮助公司设计客户需求网站或系统的架构，并设计出项目方案。
- 参与项目实施，按先前的项目方案完成，直至客户签收。
- 搭建内网开发测试环境，确保公司开发和测试人员的工作顺利进行。

在进公司前本来说好是做 Linux/Unix 系统管理员的，结果到公司以后发现原来是项目实施，工程师所幸做的项目清一色是 Linux 集群相关的，又有大量的项目实践机会接触 F5、LVS/HAProxy，还可以替客户设计其网站架构，所以工作还是蛮愉快的。这段时间各方面的技术能力增长不少，非常有成就感。

现在所从事的职业是一家外企的 Linux/Unix 系统管理员，主要职责是保证公司的电子商务网站运行顺利，其工作职责也有 3 点：

- 升级及监控维护公司电子网站的集群架构，保证其 365 × 24 小时无故障运行（这一点感觉有些辛苦，手机 24 小时开机，有时凌晨 2:00 多要起来处理网站紧急故障）。
- 搭建内网（FreeBSD8 + ezjail）开发环境，复制线上网站的 Web 环境（Apache + PHP + MySQL）供开发人员调试，保证其工作顺利。

- 因为公司网站业务也涉及开发人员，所以需要配置 OpenVPN 移动办公环境。配置这个环境时有点不顺利，初期采用的是 OpenVPN 路由模式，后期考虑到公司的网络环境而改用 OpenVPN 的网桥完成了此项工作。

总体说来，我发现系统管理员所需掌握的知识要非常全面，工作中的技术含量非常高，有服务器操作系统（比如现在较流行的 Linux 和 FreeBSD 系统）、数据库（MySQL 和 Oracle 数据库）、网络设备调试（防火墙、路由器和交换机）、代码开发（PHP、Java）等，我们需要根据公司的侧重点不同而投入不同的精力。另外，系统管理员现在主要的工作重心还是系统运维（这也是最有技术含量的），即保证公司外围网站稳定运行，因为涉及的是公司的核心业务，比如电子商务网站或 CDN 广告网站等，所以需要掌握的技术包括网络安全（包括 Linux 防火墙和硬件防火墙）、负载均衡高可用、CDN 相关技术、存储、MySQL 或 Oracle 数据库，并且要非常熟悉 SHELL 或 Python，还要懂 PHP 或 Java 语言等。相对而言，其技术含量是高于一般网络工程师和开发人员的。

10.2 系统管理员应该熟悉的系统

许多刚从事系统维护的朋友或正在学习系统相关知识的同学可能会很迷惘，现在有这么多的服务器版本（尤其是 Linux），该如何学习并深入研究下去呢？是不是只学习 Windows Server 系列或 Linux/Unix 即可？大家不要着急，相信在看完以下内容后，你会找到答案。

首先纠正一下系统管理员（System Admin）容易产生的 3 个误区：

- 究竟以 Windows Server 做服务器好还是以 Linux/Unix 做服务器好？

我以前的同事老爱争论这个问题，其实我觉得完全没必要。对于你的公司而言，Windows Server 2003 好，那就用 Windows Server 2003；Linux 好就用 Linux，谁安全和高效就用谁。当然了，不建议用盗版。如果是做域控或桌面机系统，建议大家用 Windows Server 2003 和 Windows XP，当然如果想要酷，也可以用 Ubuntu8.10，但要保证不影响工作，并且可满足办公需求，Ubuntu8.10 加 Windows Server 2003 域控并用 Exchange2007 收发工作邮件不是件容易的事。

- 系统管理员应该了解网络知识，许多人容易忽视这个问题。

虽然在大型公司中，系统管理员和网络管理员是明显不同的两个工种，但在许多工作中，例如网站维护和开发环境部署是需要双方协同合作的，缺一不可。我觉得作为 System Admin，应该掌握和了解市场上防火墙的性能与特点，同时还要掌握一些简单的交换机和路由机的配置及 iptables 或 VPN 的相关知识。或许可以对网络不甚精通，但一定要熟悉和了解，如果网站出了问题，我们可以在第一时间判断是网络还是系统的问题，然后有针对性地排障。

- 系统管理员应了解开发编程语言。

先不说 Windows 下的批处理和 Linux 下的 SHELL 编程，这应该是系统管理员必须掌握的基本功。我说的是 PHP 或 Java，还有 C 语言，系统管理员管理的是应用层的东西，了解这些编程知识会对自己很有好处，如果你将来想要成为系统架构师，那么它们就是铺垫和基础。开发是应用最底层的东西，我认识的系统总监无一不是写代码出身的。如果有精力，可适当加强这些方面的学习，个人比较推荐的语言是 PHP，入门简单，而且它也是一门强大的网站开发语言。

下面介绍一下 Windows Server 系列，我接触 Windows Server 系列比较早，记得是在 2002 年，我大学毕业后在某大型国企信息技术部上班，负责武汉地区各门市店的进销存数据。当时值守的是 Windows Server 2000 服务器，把它放在内网作 FTP 文件服务器还可以，但如果放到公网上而且忘了

打补丁就是悲剧了，冲击波和震荡波病毒会搞得你崩溃。所以后来每次上新服务器的新系统时，我们会按照计算机总部发的补丁列表老老实实一个个地打补丁，这种情况一直到后来用了 Windows Server 2003 后才有所改善。说老实话，Windows Server 2003 是我非常喜欢的一个系统，如果作为域控或 Exchange2007 服务器，我都推荐用它。此外在 Windows Server 2003 上安装 SQL Server 2008，也非常稳定，但这些都是收费的，而且价格不菲。我比较喜欢在内网测试环境下，在 Windows Server 2003 服务器上部署 VMware Server1.01 给开发人员使用，效果相当好，一台组装的 i5（8G 内存）可以虚拟大约 8 套系统，足够一个测试小组使用了。

另外，建议大家在工作中以 64 位的 Windows Server 2003 为主，这是因为：相比于 32 位的 Windows 系统，64 位的 Windows Server 2003 数据中心版最多可支持 64 路处理器，而 32 位版本只支持 32 路处理器。而 64 位的 Windows Server 2003 在内存支持方面的提高也颇为显著，32 位的 Windows 2003 最多只能支持 64GB 的内存，但是 64 位的大部分版本可支持多达 1TB 的内存。也许现在大部分个人用户都还没有用上超过 1GB 的内存，但是在高阶运算领域，扩展能力不够是 32 位系统无法被应用的主要原因之一。在包括系统模拟、基因运算等很多大运算量的任务中，一个运算进程可能要占用几个 GB 的内存，而每一次运算任务都会产生十几个 GB 的数据。所以我的生产环境上的 Windows Server 2003 基本都是 64 位的。Windows Server 2003 不难学，建议大家熟练掌握之。

另外，Windows Server 2003 的升级版 Windows Server 2008 既有细节的变化，也有根本性的变化。服务器整合、硬件的高效管理、远程的硬件无图形界面操控，彻底改变的系统安全模式，这些都是 Windows Server 2008 的大卖点。我比较关心的有以下几点：

- 具有自修复功能的 NTFS 文件系统（the self-healing NTFS file system）。在 Windows Server 2008 中，有一个新增的系统服务在后台检测文件系统的错误，它能在服务器运行状态下进行直接修复。如果检测服务正在修复损坏的磁盘结构，对应用程序来讲，受到的影响只是存储在这些区域上的文件暂时无法访问，在修复结束后即可重新访问。系统是永远不会关闭的，没有必要通过重启来做 CHKDSK 这样的操作。
- SMB2 网络文件系统（SMB2 network file system）。在很久以前 SMB 就成为了 Windows 自带的网络文件系统。现在随着多媒体文件体积与日俱增，对服务器的要求也相应地提高了。在微软的内部测试中 SMB2 的速度比 Windows Server 2003 中的网络文件系统要快 3 到 4 倍。
- 全新的命令行工具（PowerShell）。PowerShell 是一种命令行界面和脚本语言，是专门为系统管理而设计的。Windows PowerShell 构建在 Microsoft.NET Framework 之上，可帮助 IT 专业人士控制和自动管理 Windows 操作系统，以及在 Windows 上运行的应用程序。微软公司早在 2006 年的第四季度就正式发布了一款基于对象的 SHELL——Powershell 2.0，它也将在 Windows Server 2008 R2 正式发布的时候完全融入到这个软件中。Windows Server 2008 R2 包括一系列新的服务器管理界面，这些均建立在 PowerShell 2.0 之上。它新增了 240 个 cmdlets 命令集，新的 PowerShell 图形用户界面也增添了开发功能，从而能更简单地创建自己的命令行。而且，PowerShell 将能够安装到 Windows 服务器内核中，PowerShell 的新特点大家可以参考具体资料：http://articles.e-works.net.cn/pc_server/article88176.htm。
- 增加了新的虚拟化软件 Hyper-V。Hyper-V 2.0 虚拟机在性能上对逻辑处理器和内存的支持得到了增强，目前的 Hyper-V 可以支持 24 个逻辑处理器，而 Hyper-V 2.0 中每个虚拟机可以支持 32 个逻辑处理器和最高 64GB 的内存。Hyper-V 2.0 的一大重要功能就是 Live Migration

(动态迁移), 配合 AMD Opteron 等处理器可以让虚拟机在开机服务状态下实现平台间的实时迁移。

Windows Server 2008 在 Windows Server 2003 的基础上增加了许多新的亮点, 虽然它目前在国内还没有普及 (这应该要归功于 Windows Server 2003 的稳定和实用), 但我们可以抱着研究的目的来学习这个新的服务器操作系统, 以便将来能在企业的应用中发挥它的长处, 摒弃它的缺点, 这也是很有成就感的事情。

再说一下 Linux 服务器操作系统。严格来讲, Linux 这个词本身只表示 Linux 内核, 但实际上人们已经习惯了用 Linux 来形容整个基于 Linux 内核, 并且使用 GNU 工程的各种工具和数据库的操作系统 (也被称为 GNU/Linux)。基于这些组件的 Linux 软件被称为 Linux 发行版。一般来讲, 一个 Linux 发行套件包含大量的软件, 比如软件开发工具、数据库、Web 服务器 (例如 Apache、Nginx)、X Window、桌面环境 (比如 GNOME 和 KDE)、办公套件 (比如 OpenOffice.org) 等。

Linux 内核最初是为英特尔 386 微处理器设计的。现在 Linux 内核支持从个人计算机到大型主机甚至包括嵌入式系统在内的各种硬件设备。在开始的时候, Linux 只是个人狂热爱好的一种产物。但是现在, Linux 已经成为了一种受到广泛关注和支持的操作系统。包括 IBM 和惠普在内的一些计算机业巨头也开始支持 Linux 了。很多人认为, 和其他的商用 Unix 系统及微软的 Windows Server 相比, 作为自由软件的 Linux 具有成本低、安全性高、更加可信赖的优势。Linux 最成功的系统莫过于 RedHat 和 Centos 了, 开源免费的 Centos 系统现在越来越受欢迎。Centos 完全是基于 RedHat 的 RHEL (RedHat Enterprise Linux) 企业发行版, 仅仅是移除了 RedHat 的商标而已。一位 Centos 用户称, RedHat 的企业服务合同价格太高了, 而且他们也不认为支付的支持服务费用是物有所值。在桌面市场上, Centos 并不算是一个流行的 Linux 发行版, 在各种热门的 Linux 发行版的排行榜上也没有特别好的排名。Centos 是一个丝毫没有个性的发行版, 它根本就是和 RedHat 企业级的 Linux 一模一样的。虽然如此, 但是在作为 Web 服务器运行的 Linux 当中, Centos 却是毫无悬念的 No. 1。根据国外科技网站 W3Techs 的数据显示, 截至 2010 年 7 月 28 日, Centos 以高达 31.6% 的份额占据了 Linux Web 服务器的榜首, 第二和第三分别是 Debian 和 RedHat。也就是说, 在每 10 个基于 Linux 的 Web 服务器当中, 就有 3 个 Centos, 两个半 Debian 和一个半 RHEL。其中, RedHat 由红帽公司开发并提供技术支持, Debian 和 Centos 都是由社区维护的。

我们可以分析一下 Centos 为什么这么流行。

为什么企业宁愿选择没有技术支持的 Centos, 而不去选择技术背景强大、支持有保障的 RedHat? 第一个原因当然是成本。在美国, RedHat 的价位大约是每个服务器每年 1000 美元; 在国内的话, 在大规模的企业中更是到了每台服务器每年上万甚至数十万元的水平, 这并不是一般企业愿意承担的。而 Centos 则相反, 它是完全免费的。另一方面, Centos 在数据中心里很受欢迎, 因为它非常容易架设, 非常容易维护和管理, 并且非常快。对于现在非常流行的高可用软件 Heartbeat 来说, Centos 是唯一一款支持直接用 yum 来安装的 Linux 操作系统。如果其他版本的 Linux 系统也想安装 Heartbeat, 只能通过源码编译的方式了。

最后, 让我们看看 W3Techs 统计数据的一些细节:

- Centos 的占有率上升主要是从 RedHat 和 Fedora 那里夺取了市场份额, 但同时, Linux 市场也有部分份额被 Ubuntu 所取代。
- Centos 在大型企业中使用得相对少, 主要还是分布在大多数中小规模的站点上。

□ 在 Centos 上使用的主要是 Sun 的服务器技术，使用 Nginx 的不多。

□ Centos 在日本、西班牙和罗马尼亚的市场份额相对较高，在德国、中国和巴西则相对较低。

现在在北京、上海及深圳，包括武汉的机房里都可以看到 Centos 的身影。如果你对开源系统非常感兴趣，我建议将 Centos 作为第一个系统来研究。

最后再说一下 FreeBSD。这也是一款性能优秀，并且稳定性在开源系统里也数一数二的服务器操作系统。

FreeBSD 是一种 UNIX 操作系统，是由 BSD、386BSD 和 4.4BSD 发展而来的 Unix 的一个重要分支。它支持 x86 兼容（包括 Pentium 和 Athlon）、AMD64 兼容（包括 Opteron、Athlon 64 和 EM64T）、Alpha/AXP、IA-64、PC-98 及 UltraSPARC 架构的计算机。它运行在 Intel x86 family 兼容处理器、DEC Alpha、Sun 微系统的 UltraSPARC、Itanium (IA-64) 和 AMD64 处理器上。针对 PowerPC 的支持正在开发中。业界人士普遍认为它是相当可靠和健壮的，苹果电脑的 Mac OS X 以 Mach 为内核，配合 FreeBSD 的驱动程序和实用工具为基础来开发。FreeBSD 源于 BSD——美国加州大学伯克利分校开发的 Unix 版本，它是由来自世界各地的志愿者开发和维护的，FreeBSD 为不同架构的计算机系统提供了不同程度的支持。

而从企业的角度来看，公司许多重要的服务器应用，都是用 FreeBSD8.1 x86_64 来运行的（FreeBSD8 系列的稳定性有目共睹），比如 postfix 和 SVN。如果大家也像我一样维护内网开发环境就会明白，熟练掌握 FreeBSD 是一件多么惬意的事情。如果开发人员需要大量的虚拟环境，这时候不需要考虑什么 VMware ESXi 和 XEN 了，直接采用 FreeBSD 自带的 jail 虚拟机即可，安装及部署都非常方便，在 DELL PowerEdge R710 上运行 10 个虚拟机都没什么问题，基本上可以满足几个开发小组的需求。而在基于 Apache + PHP5 + MySQL 的开发环境中，用 FreeBSD8.1 x86_64 就是一种享受。相对于外网源码安装的 LAMP 和 LNMP 环境来说，二者所投入的精力和时间就不在一个等级上。诸如 Samba、vsftpd 和 SVN 等这些企业内部常见的文件服务器环境和开发环境，一个 port 就可以很轻松地解决了。在 FreeBSD8.1 上安装 SVN 确实比在 Linux 下安装 SVN 要简单得多，在安装重要的 OpenVPN 服务器时，经过对比，我还是选择了安装在 FreeBSD8 x86_64 系统上。我现在比较倾向的做法是：外网环境（需要部署 Nginx 和 Heartbeat 的网站）考虑用 Centos5.5 x86_64，而内网开发环境，我会尽量使用 FreeBSD x86_64。

有关 Linux 和 FreeBSD 到底谁最好的争论一直都没停过。其实谁好谁稳定都只是暂时的，我认为这是一个“既生瑜何生亮”的问题，在长久的发展过程中，技术上的常胜将军并不存在，也许只有此消彼长，各领风骚，这才会让竞争更健康。至于大家为什么非要争论某一个更好，我想程序员普遍都喜欢追求完美，非要用最好最完美的系统才甘心吧！其实，这两个操作系统都可以熟悉和了解一下，取它们所长，按其短是一种很好的做法。

Windows Server 2003 (Server 2008)、Centos、FreeBSD 这 3 种系统各有各的特点，大家可根据公司的需求决定投入的精力和时间。另外，也可以顺便研究一下 Debian6（我认为它是操作系统中最智能的一个，一个 apt-get install 可以解决 60% 的问题），至于 AIX 和 Sun 的 Solaris，有精力和机会的朋友可以研究和深入了解一下。系统只是一个载体，谁能够最大限度地发挥网站的整体性能，降低企业成本，就优先考虑用谁。

参考文档：

<http://os.51cto.com/art/201007/214488.htm>

http://articles.e-works.net.cn/pc_server/article88176.htm

10.3 系统管理员应该熟悉的工具

系统工程师远程管理 Linux/UNIX 服务器时，并不是直接在 Linux/Unix 上通过 SSH 远程管理服务器的。系统管理员也只是一名普通的员工，平时在 Windows XP 或 WIN7 下通过 Office Word 2007 写文档和报告，通过 Office Outlook 2007 收发电子邮件。所以，对于服务器的管理，也是在 Windows 操作系统下通过工具来远程管理的。下面给大家推荐一些相应的工具，我平时都是用它们来进行管理的。大家可以先熟悉，然后再试用，最后通过实践掌握其用法，达到提高工作效率的目的。

1. PieTTY

PieTTY 是由林弘德 (Hung-Te Lin, piaip) 以 PuTTY 源代码为基础，在 Windows 上发展的 Telnet/SSH 安全远端连线路程，修正并且完整支援亚洲语系字符，可切换多种 Unicode 字符显示方式，提供简易 SCP 上传界面，并增加了透明视窗、无边框模式等视觉效果。PieTTY 与 PuTTY 同样采用 MIT License，但 PieTTY 目前并没有释出源代码，如果您有非常高度的安全需求，请自行斟酌是否使用 PieTTY。PieTTY 使用 MIT License，是免费的，而且可自由发布。

Putty 是个小巧方便的 Telnet/SSH 安全远端连线路程，但用于非英语系文字时有非常多的问题，而且对于初学者来说，它过于复杂的使用界面也为人诟病已久。并在其使用界面上大幅改进、易学易用的版本。PieTTY 0.3 系列是修改自 PuTTY 0.57/0.58 的版本，以稳定与修正为主。其主要特色如下：

- 简单易用的界面（中英文合一），主要的功能都可从选单中选取。
- 完全相容于传统的 PuTTY，之前的设定全部可直接使用。
- 更强的连线整合管理（session management），自动储存设定。
- 高度可自订化（customizable）的视窗显示效果。
- 完整而方便切换的多国语言支援。
- 半透明显示（多种显示引擎以配合各种硬件配备与视窗立体阴影，配合无框显示模式效果奇佳。
- 对于各种网址 URL 可直接点选开启，还有各种可选用的视觉效果。
- 支持拖曳档案（Drag-n-drop）SCP 上传。

PieTTY 官方网站：<http://ntu.csie.org/~piaip/pietty/>

2. Screen

简单来说，Screen 是一个可以在多个进程之间多路复用物理终端的窗口管理器。Screen 中有会话的概念，用户可以在一个 Screen 会话中创建多个 Screen 窗口，在每一个 Screen 窗口中就像操作一个真实的 Telnet/SSH 连接窗口那样。Screen 还有更高级的功能。你可以不中断 Screen 窗口中程序的运行而暂时断开（detach）Screen 会话，并在随后重新连接（attach）该会话，重新控制各窗口中运行的程序。我在附录 B 中会详细说明其使用方法。

3. WinSCP

WinSCP 是一个 Windows 环境下使用 SSH 的开源图形化 SFTP 客户端。它同时支持 SCP 协议。它的主要功能就是在本地与远程计算机之间安全地复制文件。此软件还可以结合 PieTTY 来使用，

以方便用户的远程登录。其特点如下：

- 图形用户界面。
- 多语言。
- 与 Windows 完美集成（拖曳、URL、快捷方式）。
- 支持所有常用的文件操作。
- 支持基于 SSH-1、SSH-2 的 SFTP 和 SCP 协议。
- 支持批处理脚本和命令行方式。
- 多种半自动、自动的目录同步方式。
- 内置文本编辑器。
- 支持 SSH 密码、键盘交互、公钥和 Kerberos (GSS) 验证。
- 通过与 Pageant (PuTTY Agent) 集成支持各种类型的公钥验证。
- 提供 Windows Explorer 与 Norton Commander 界面。
- 可有选择地存储会话信息。
- 可将设置存在配置文件中而非注册表中，适合在移动介质上操作。

它操作文件的特点如下：

- WinSCP 可以执行所有基本的文件操作，例如下载和上传。同时允许为文件和目录重命名、改变属性、建立符号链接和快捷方式。
- 有两种可选界面允许用户管理远程或本地的文件。
- 连接到远程计算机。
- 使用 WinSCP 可以连接到一台提供 SFTP (SSH File Transfer Protocol) 或 SCP (Secure Copy Protocol) 服务的 SSH (Secure Shell) 服务器，通常是 Unix 服务器。SFTP 包含于 SSH-2 包中，SCP 在 SSH-1 包中。两种协议都能运行在以后的 SSH 版本之上。WinSCP 同时支持 SSH-1 和 SSH-2。

WinSCP 官方网址：<http://winscp.net/>

注意 建议将 PuTTY + Screen 配合 WinSCP 使用，这样不仅速度快，而且能满足工作中的大部分需求，能提高我们的工作效率。

4. Xmanager 3.0 企业版

它是 Linux/Unix 的系统管理员必备的软件之一，功能极其强大，其特点如下：

- 跟 PuTTY 和 PuTTY 不同的是，它只需要一个程序窗口就可以同时控制成百台的 Linux/Unix 服务器。
- X-browser 能很好地从 Windows 桌面控制 Linux 桌面，尤其是在 Linux 下安装 Oracle 数据库时或通过 Wireshark 抓图工具分析数据包时非常有用。
- X-sftp 可以很方便和安全地上传、下载 Linux/Unix 服务器的资料和数据，尤其是在 Windows XP 下，它支持 FTP 和 SFTP。
- 通过它的 Xshell 3.0 可以很方便地同时操控几台服务器，尤其是在部署 Web 集群和 Squid 集群环境时非常方便。

其使用方法将在附录 A 中详细介绍。

Xmanager 官方网站: <http://www.netsarang.com/products/enterprise.html>

5. FileZilla

FileZilla 是一个免费开源且跨平台的 FTP 解决方案, 分为客户端版本和服务端版本, 具备所有的 FTP 软件功能。在 Windows、Linux、Mac OS X 下均有对应的版本。软件许可证为 GPL。可控性、有条理的界面和管理多站点的简化方式使得 FileZilla 客户端版成为一个方便高效的 FTP 客户端工具。而 FileZilla Server 则是一个小巧并且可靠地支持 FTP&SFTP 的 FTP 服务器软件。我现在虽然用 Xmanager 自带的 Xftp 来代替它, 但我常推荐公司的同事使用它, 毕竟不是人人都喜欢用 Xmanager 3.0 企业版的。FileZilla 的界面非常友好和人性化, 它的界面是仿 IE 的。

FileZilla 官方网站: <http://filezilla-project.org/>

6. gVim

gVim 是 Windows 版的 Vim, 并且有标准的 Windows 风格的图形界面, 所以叫 g (graphical) Vim。这是一个国际版本, 会根据安装的平台自动选择相应语言包, 支持中文及其各种编码, 连界面也是中文的, 英文不是太好的朋友也可放心使用。相信这个极具 Unix 特色和风格 (simple is the best) 的编辑器会给大家带来不同的感受。建议大家在 Windows 系统下使用它, 编辑 PHP (Java) 文件、SHELL 文件、.conf 服务配置文件和 OpenVPN 的拨号文件都非常方便。

gVim 官方网站: <http://www.vim.org/>

7. 远程桌面、PCanywhere、Remote Admin

这些是控制 Windows Server 2003 及 Windows Server 2008 服务器远程桌面的软件, 相信大家对远程桌面和 PCanywhere 已经比较熟悉了, 这里就重点介绍一下后面的 Remote Admin, 我现在主要用其控制公网的 Windows Server 2003 机器。

Remote Admin 有以下特点:

- ☐ 运行速度快。
- ☐ Remote Admin 支持被控端以服务的方式运行, 支持多个连接和 IP 过滤 (即允许特定的 IP 控制远端机器)、个性化的文件互传、远程关机, 支持高分辨率模式、基于 Windows NT 的安全支持及密码保护, 提供日志文件的支持等。
- ☐ 在安全性方面, Remote Admin 支持 Windows NT/2000 用户级安全特性, 你可以将远程控制的权限授予特定的用户或用户组, Remote Admin 将以加密的模式工作, 所有的数据 (包括屏幕影像、鼠标和键盘的移动) 都使用 128 位强加密算法加密。服务器端会将所有的操作写进日志文件中, 以便于事后查询。服务器端有 IP 过滤表, 对 IP 过滤表以外的控制请求不予响应。
- ☐ Remote Admin 目前支持 TCP/IP 协议, 应用十分广泛。

Remote Admin 官方网站: <http://www.radmin.com/>

8. 服务器监控软件 Cacti 和 Nagios

这两个软件的网上资料非常多, 建议大家熟悉掌握它们, 尤其是 Nagios, 它现在的邮件/短信报警做得相当好, 只要你手机 24 小时开机, 如果遇到服务器服务死掉或服务器宕机的情况, 它会非常负责任地不间断地 (请注意这个字眼) 提醒你服务出问题了, 直到你去处理为止。

Cacti 官方网站: <http://www.cacti.net/>

Nagios 官方网站: <http://www.nagios.org/>

9. 扫描之王——Nmap

在 Windows Server 2003 和 Linux/Unix 上都有其对应的版本, 这一款免费开源的安全工具大家都应该了解和掌握。

Nmap 官方网站: <http://nmap.org/>

10. 抓包工具 Wireshark 和 TCPDump

Windows Server 下的抓包工具 sniffer 大家都应该有所接触, 这里重点推荐一下 Linux 下的 Wireshark, 它界面友好, 语法简单, 是我在 Linux 下进行抓包工作的首选工具。如果对命令行非常熟悉, 可以考虑用命令 TCPDump 来抓包。

Wireshark 官方网站: <http://www.wireshark.org/>

11. 防火墙工具 iptables

FreeBSD、OpenBSD 下的防火墙是 ipfw, Linux 下是 iptables, 个人重点推荐 iptables, 它的语法简单易学, 作 NAT 路由器也很方便, 大家可以重点关注第 7 章的相关内容。

12. 数据库工具——phpMyadmin

作为 Linux/Unix 系统管理员也免不了要熟悉 MySQL 数据库。虽然有 phpMyadmin 这个好工具帮我们管理 MySQL 数据库, 但建议大家还是熟练掌握 MySQL 的命令语法, 毕竟不可能你的每台 MySQL 数据库都配有 phpMyadmin, 尤其是在运行 Java 应用环境的机器上。

phpMyadmin 官方网站: http://www.phpmyadmin.net/home_page/index.php

13. MySQL 图形化工具 SQLyog

SQLyog 是业界著名的 Webyog 公司出品的一款简洁高效、功能强大的图形化 MySQL 数据库管理工具。使用 SQLyog 可以快速直观地让您从世界的任何角落通过网络来维护远端的 MySQL 数据库。

SQLyog 相比于其他类似的 MySQL 数据库管理工具, 有如下特点:

- ☐ 基于 C++ 和 MySQL API 编程。
- ☐ 方便快捷的数据库同步与数据库结构同步工具。
- ☐ 易用的数据库、数据表备份与还原功能。
- ☐ 支持导入与导出 XML、HTML、CSV 等多种格式的数据。
- ☐ 直接运行批量 SQL 脚本文件, 速度极快。
- ☐ 新版本更是增加了强大的数据迁徙功能。

14. 虚拟机软件系列

基于系统级别的虚拟机软件, 推荐 Linux 下的 XEN 和 FreeBSD 下的 jail, 详细内容大家可以参考前面第 2 章和第 3 章的内容。

专业的虚拟化软件我推荐 VMware ESXi 和 Citrix 的 XenServer 5.6。

VMware ESX 服务器是在通用环境下分区和整合系统的虚拟主机软件的。它是具有高级资源管理功能、高效、灵活的虚拟主机平台。VMware ESX Server 为适合任何系统环境的企业级虚拟计算机软件。大型机级别的架构提供了空前的性能和操作控制。它能提供完全动态的资源可测量控制, 适合各种要求严格的应用程序的需求, 同时可以实现服务器部署整合, 为企业未来成长所需扩展空

间。它可以以 VMotion 技术在各服务器或刀片服务器之间弹性动态迁移系统平台，让 IT 人员做更有效的资源调度，并获得更好且安全周密的防护。当系统发生灾难时，可以在最短的时间内迅速复原系统的运作（无须重新安装操作系统），推荐大家用免费版本 VMware ESXi 4.1。

服务器全虚拟化产品 Citrix 的 XenServer 源自于开放原始码的 Xen。和大多数服务器半虚拟化产品相同的是，XenServer 作为一种开放的、功能强大的服务器虚拟化解决方案，可将静态的、复杂的数据中心环境转变成更为动态的、更易于管理的交付中心，从而大大降低数据中心成本。推荐大家用 XenServer 5.6 搭建自己的学习测试环境，它的兼容性非常好，只要台式机的 CPU 是基于 64 位的，安装成功的几率就会非常高。

15. 分布式管理 Linux 主机工具 Puppet

Puppet 是一个可以管理大量 Linux 主机的系统，它的中心思想是让主机上的某个资源及配置文件（manifest）与该资源的设计一致，这里的资源可以指一个文件、一个用户、一个软件包、一条 cron 中的任务等实体。开发 Puppet 是为了让系统管理员社区可以相互交流和共享成熟的工具，避免重复的劳动，这里只简单介绍一下，在附录 C 中我会详细说明其语法特点和用法。

10.4 Linux 的学习及进阶之路

随着 Linux 应用的发展，有更多朋友开始接触 Linux 了，但是在根据学习 Windows 的经验来学习 Linux 时往往有茫然的感觉，不知从何处开始学起。作为一个 Linux/Unix 系统管理员，我看了许多有关 Linux 的文档和书籍，并为学习 Linux 付出了许多艰苦的努力。Linux 相关的系统知识博大精深，但是只要掌握了重点知识，让自己的能力提高到一定程度是没有问题的。下面我将这方面的一些工作心得介绍给大家。

1. 学习目的

在这个网络人才身价倍增的年代，想靠技术吃饭又不想掌握网络和编程技术是不明智的。当大家第一次听说 Linux 并跃跃欲试的时候，总会提出几个问题：它是什么？为什么要用它？怎样学习它？作为开放源码运动的主要组成部分，Linux 的应用越来越广泛，从我们平时的娱乐、学习，到商业、政府办公，再到大规模集群的应用。为了满足人们的各种需求，各种各样基于 Linux 的应用软件层出不穷。只要具备了 Linux 的基本功，并具有了自学的能力之后，任何人都可以通过长期的学习掌握相关的专业内容。

2. 从命令开始，从基础开始

有些朋友一接触 Linux 时就是希望构架网站，根本没有想过要先了解一下 Linux 的基础知识，了解一下它的命令。虽然 Linux 桌面应用发展得很快，但是命令在 Linux 中依然有很强的生命力。Linux 是一个命令行组成的操作系统，其精髓就在命令行，无论图形化界面发展到什么水平，这个原理是不会变的。Linux 命令有许多强大的功能，如简单的磁盘操作、文件存取、复杂的多媒体图像和流媒体文件的制作等。下面把其中比较重要的和使用频率最多的命令，按照其在系统中的作用分成几个部分介绍给大家，通过这些基础命令的学习我们可以进一步理解 Linux 系统。

- 安装和登录命令：login、shutdown、halt、reboot、mount、umount、chsh
- 文件处理命令：file、mkdir、grep、dd、find、mv、ls、diff、cat、ln
- 系统管理相关命令：df、top、free、quota、at、lp、adduser、groupadd kill、crontab、tar、

unzip、gunzip、last

- 网络操作命令：ifconfig、ip、ping、netstat、telnet、ftp、route、rlogin rcp、finger、mail、nslookup
- 系统安全相关命令：passwd、su、umask、chgrp、chmod、chown、chattr、sudo、pswho

3. 搭建合适的 Linux/Unix 学习平台

有很多朋友或同学向我抱怨，说没有服务器环境安装 Linux 系统进行相应的学习和工作。这应该是很少做实验的缘故。我们先说一下比较专业的虚拟化软件 XenServer5.6，它完全可以在一台双核速龙+2G 内存的机器上虚拟出 4 个 Linux 系统来，32 位或 64 位均可，再搭配一个几十元的家用路由器，就可以成为一个局域网了。XenServer5.6 的兼容性非常好，在 64 位 CPU 的台式机上基本都能安装成功，大家可以尝试一下。有的朋友可能会说这样浪费了一台双核主机，其实并不会，我们完全可以用 VMware Server 来做此事。也可能会有朋友有这样的疑问：我在速龙主机装 32 位的 Windows XP，然后再装 Centos5.5 x86_64，这样可行吗？我可以很肯定地告诉大家，这完全是可行的。有些朋友可能还是学生，一般是用手提电脑在宿舍上网的，这也有解决办法，完全可以利用 VMware Workstation 虚拟一台 Linux 机器出来专作学习之用。建议大家平时多动手多做实验，毕竟实践出真知。

4. 选择好的 Linux 书籍

无论是在论坛还是在读者反馈中，我们看到的最多的问题往往是新手针对安装或使用 Linux 的过程中遇到的一个具体问题而进行的提问，其中有很多都是重复性的问题，甚至有不少人连基本的问题都描述不清楚。这说明很多初学 Linux 的人还没有掌握基本功。怎样才能快速提高掌握 Linux 的基本功呢？最有效的方法莫过于学习权威的 Linux 工具书了，工具书对于学习者而言是相当重要的。不过，一本观念错误的工具书却会让新手误入歧途。我现在比较喜欢看的书籍和文档都是基于真正的线上环境的，希望大家在选择书籍时注意甄别。这里还有一个小心得：平时我们手边或床上应该放几本学习的书籍，有事没事时都可以翻着看看，你会发现许多内容会在不知不觉中就被吸收进去了，而且这种方法比平时专门看书学习的效率要高很多，大家不妨试一试。

5. 用 Unix 的思维思考 Linux

由于 Linux 是参照 Unix 的思想来设计的，理解和掌握它就必须以 Unix 的思维来进行，而不能以 Windows 的思维来思考。不可否认，Windows 在市场上的成功很大一部分在于技术思想有其独到之处。可是这个创新是在面对个人用户的前提下进行的，而面对企业级的服务应用，它还是有些力不从心的。多年来在计算机操作系统领域一直是二者独大：Unix 在服务器领域，Windows 在个人用户领域。由此可见，用户需求决定了所采用的操作系统。不管什么原因，如果要学习 Linux，那么首先要将思维从 Windows 中拉出来，转而以 Unix 的思维方式来思考。比如，Unix 基本哲学之一——一切皆文件，要真正完全理解和掌握这句话的含义，也只有在工作中才能实现。

6. 养成在命令行下工作的习惯

一定要养成在命令行下工作的习惯，要知道 X Window 只是运行在命令行模式下的一个应用程序。在命令行下学习虽然一开始进度较慢，但是在熟悉后，大家学习之路将以指数增长方式增长的。命令行实际上就是规则，它总是有效的，同时也是灵活的。即使是通过一条缓慢的调制解调器线路，它也能操纵几千公里以外的远程系统。现在专业的系统管理员都能通过命令，操纵上千台 Linux/Unix 机器，但如果是图形化界面操作，显然达不到这种效率。

7. 学习 SHELL

对于 SHELL (中文名为壳), 习惯 Windows 的读者肯定会觉得非常陌生, 因为 Windows 只有一个“Shell”(如果可以说是 Shell 的话), 那就是 Windows 自己。用一句话来解释就是, SHELL 是用户输入命令与系统解释命令之间的中介。最直观的说法是, 一种 SHELL 有一套自己的命令。举一个容易理解的例子, Linux 的标准 SHELL 是 bash, Linux 的 SHELL 是以命令行的方式表现出来的。有的读者可能会不理解, Windows 从命令行“进化”到了图形界面, 那么 Linux 现在还使用命令行岂不是一种倒退? 当初我刚刚接触 Linux 时就有过这种想法。可是后来发现, 如果使用图形界面, 那么分配给应用程序的资源就少了, 在价格昂贵的服务器上, 能够以较低的硬件配置实现同样的功能是非常重要的。下面来举例说明, 一台服务器有 1GB 内存, 假设其中有 512MB 用于处理图形界面, 若要安装一个需要 784MB 内存的数据库软件, 唯一的办法就是扩大内存。但是如果使用命令行, 系统可能只需要 64MB 内存, 其他的内存就可以供数据库软件使用了。使用命令行, 不仅是内存, 在 CPU 及硬盘等资源的占用上都会节省很多, 这也是我推荐大家在安装 Linux 服务器时采用最小化安装的原因, 它是没有图形化界面的。所以, 作为服务器, 使用命令行是优点而不是缺点。既然 SHELL 有这么多优点, 学习和使用它则是理所当然的, 虽然学习过程可能会很枯燥。大家可以用我在学习 Python 的方法来学习 SHELL, 我每天规定自己学习 10 行代码或一个知识点, 这样日积月累也是相当可观的。学习 SHELL 是一个循序渐进的过程, 只要坚持, 很快就会发现我们也能在 Linux 下熟练地用 SHELL 进行工作了。

8. 勤于实践

要增加自己 Linux 的技能, 只有通过实践来实现了。大家可以根据我前面所推荐的虚拟化软件, 安装一个 Linux 发行版本, 然后进入精彩的 Linux 世界, 相信会大有斩获的。如果你无法经常学习和实践的话, 很容易学了后面的, 就忘了前面的。如果你对 Linux 命令熟悉后可以搭建一个小的 Linux 网络, 这是最好的实践方法。Linux 是网络的代名词, Linux 网络服务功能非常强大, 不论是邮件服务器、Web 服务器、DNS 服务器等都非常完善。当然你不需要搭建所有服务, 可以慢慢来。做技术不能人云亦云, 但一定要要多动手, 少提问, 自己在实践中掌握的知识 and 心得就是自己的本领了。

9. 学会使用文档和管理文档

和私有操作系统不同, 各个 Linux 发行版本的技术支持时间都较短, 这对于 Linux 初学者来说往往是不够的。其实如果你安装了一个完整的 Linux 系统, 其中就已经包含了一个强大的帮助, 只是你可能还没有发现和使用它们的技巧。主流 Linux 发行版都自带非常详细的文档(包括手册页和 FAQ), 从系统安装到系统安全, 针对不同层次的人都有详尽的文档, 仔细阅读文档后, 40% 的问题都可解决。查阅经典工具书和 Howto 又可以解决 40% 的问题, 特别是 Howto, 它是全球数以万计的 Linux、Unix 的经验总结, 非常有参考价值。安装一个新的软件时先看 README, 再看 INSTALL, 然后看 FAQ, 最后再动手安装, 这样当遇到问题时就知道为什么了。如果不看说明文档, 出了问题就去论坛找答案, 这反而会浪费时间。当查找文档时, 一定要看是在何种版本、何种环境以及何种状态下出现的结果。对于文档的有效性, 一时无法在操作前就知道结论如何, 那么对某个专题或问题, 阅读相关的多篇文章将会节省大量的时间, 还可以保证尽量“干净”的环境, 有效避免因为不同操作所造成的更多问题。操作时要仔细核对各个步骤及输出的结果, 尽量保持与文档一致。在

工作中,我发现有一个方法效率非常高,而且对别人的帮助也非常大,那就是把我的工作过程,包括操作流程和作用心得形成文档并流程化,这样对同事们和自己以后的工作也是非常有帮助的。现在文档化在很多公司都已经成为一种制度了。

10. 在 Linux 论坛上获取帮助

如果上面的方法都没有解决问题,此时就需要 Linux 社区的帮助了。需要说明的是,我们在提问前要有周全的思考,准备好问题,不要草率发问,否则只会得到草率的回答,或者根本得不到任何答案。在寻求帮助前为解决问题所付出的努力越多,就越能得到实质性的帮助。最好先搜寻一下论坛中是否有您需要的文章,这样可以获得事半功倍的效果。有时我们可能会遇到这种情况,对于同一个问题会出现不同的回答,这时我们就需要通过实践来验证了。另外把这个问题放在其他 Linux 社区请求帮助也是一种选择。如果在某一社区得不到答案,请不要以为大家无法帮助你,有时只是看到问题的人不知道答案罢了,这时换一个社区是不错的选择。

11. 在社区共享你的经验

随着 Linux 应用的扩展,出现了不少 Linux 社区。其中有一些非常优秀的社区,比如: <http://www.chinaunix.net/> (中国最大的 Unix 技术社区)。你可以随着知识的积累,把自己动手解决的一些前人没有遇到的问题的过程写成文档放在网络上共享。这个时候,你也就成为了一名“高手”。Linux 的使用者一般都是专业人士,他们有着很好的电脑背景且愿意协助他人, Linux 高手更具有鼓励新手的文化精神。

12. 在项目中提高自己的 Linux 水平

如果以上各点你都做到了,那么还可以多做一些工作,比如加入公司的业务团队,跟开发人员和测试人员一起为项目做准备、调试、实施、维护,这时候我们会发现,许多理论知识在项目中都得到了体现,这样,我们的 Linux 水平会精进到另一个层次,我们不仅会思考一种服务或几台服务器的相关状况,我们还会考虑更多,比如服务器架构,网站调优,网站的扩展性的设计方法、数据库压力过大问题、配置分布式缓存的必要性等。

10.5 系统管理员应该如何工作

大多数企业并不愿意在 IT 支持方面额外投入过多的资金和人力,高层更愿意把资金投到销售等见效快的渠道上去,所以系统管理员小组长期存在的问题就是时间紧和资金少。这个时候,我们应该怎么合理地分配我们的资源,在高效工作的同时,也为企业节约成本呢?

1) 合理使用脚本实行任务自动化。事实上,系统管理员的日常工作大多数是数据库的备份、日志的轮询处理、定时清理磁盘等,我们完全可以利用 SHELL 来定时执行这些任务,从而争取更多的时间来处理突发事件和疑难问题。在日常工作中,我们可以借助脚本自动化充分地利用现有的资源来完成工作。一旦做到这一点,系统组可以改善自己在他人心目中的看法和形象,从而争取更多的资源,把 80% 的工作交给脚本处理,这应该也是我们工作中努力实现的一个目标。

2) 在企业中尽量使用开源系统和软件,节约企业成本。当老板或财务总监抱怨 IT 的成本预算过多时,我们可以尽量地使用开源的系统和软件来节约预算成本。大家可以对比一下 10 台 Windows Server 2008 + SQL Server 2008 与 10 台 Centos5.5 + MySQL 的成本预算,前者一个金额庞大,而后者几乎是零,可前者能实现的后者一样可以实现。现在许多强大的开源软件也带动了开源系统,比如

强大的 Apache、Nginx 和 LVS，可以说有它们的地方，就有 Linux 和 Unix，这也是许多 Windows 系统管理员转向 Linux/Unix 的原因。这段时间也有许多朋友托我帮助他们进行网站的迁徙工作，他们纷纷将自己的网站由 Windows + IIS 转向了现在流行的 LNMP 架构。公网上的 DNS 服务器，基本上是清一色的 bind9 版本，而现在许多企业和网吧，也在用 iptables 来代替硬件路由器，他们觉得在成本和管理上，iptables 更胜一筹，并且合理地利用免费的 VMware ESXi 和 Citrix 的 XenServer 来做生产环境下的虚拟化，可最大限度地节约硬件成本和人力成本。很多朋友可能会觉得服务器贵，其实如果再深入了解一下服务器置于 IDC 机房的电力成本和光纤费用，要是再加上人力成本的话，那将会是一个惊人的数目，而如果利用虚拟化技术将会节约其中的大量资金成本和人力成本。

3) 关于重要的网站或业务系统，我建议用 LVS 等 Linux 集群来实现。试想，如果公司是家跨国性质的 B2B 电子商务网站，公司业务繁忙的时候，突然电子商务挂掉了，服务器硬件损坏了，我们这时候难道手动去切换备份的网站吗？而且国外的许多电子商务网站都是欧洲时间或美国时间，如果电子商务出问题了，美国那边的同事打来紧急电话，我们再匆忙上公司的 VPN 进内网调试，会浪费许多宝贵的时间，公司的业务肯定要受到影响。如果我们的电子商务网站采用的是 Linux 集群架构，任何一台 Web 或负载均衡器出问题，都有相应的备份进行弥补，整个切换都是无人值守的，在小于一秒的时间内解决问题，这样不是更智能更人性化一些？这也是许多读者对 Linux 集群非常感兴趣的原因之一。我们也应该利用相应的技术，部署我们重要的电子商务网站或核心业务，以防万一。当然了，适当的监控也是必须的，这样可以找出网站出问题的地方，到底是硬件问题还是人为的攻击造成的。

4) 文档流程化。当我们进到一个新公司时，我们可以参考前任留下的文档，纠正他们的错误，解决历史遗留问题，迅速进入工作状态。如果你在一个公司工作很长一段时间后，也应该将工作中的过程和经验心得写成文档，并传至公司的 Wiki 服务器跟大家分享。这样进行开发的同事或测试人员也能够根据我们的文档进行相应的工作。特别是新同事上任时，我们可以根据文档进行适当的引导，迅速让其适应公司的环境，进入到工作状态中。所以文档流程化是非常重要的工作，大家都应该在自己的工作中建立相应工作的文档，代码应该写得通俗易懂，建议在涉及比较核心业务的代码编写文档中，提供一份帮助文件，详细说明此代码的作用、路径和使用方法，这样在方便自己的同时，也方便了别人。

5) 尝试和开发人员或其他人员协同工作。开发人员更了解代码资源，而我们更关心硬件和部署，有时可以在一起交叉培训和参加策略会议。如果大家能够相处得很融洽，也可以更从容地应对各种紧急事件。大家要记住的是，每个人在自己的领域都有自己的专长，我们都是为同一家公司的同一产品服务的，所以尽可能不要对别人要求过多，良好的人际关系更有利于工作。

6) 公司的网站或业务系统尽可能只做最小化的变更。如果不是必须改变的，那么就保持原样。有时候我们会面临高层的压力（比如系统总监），他们会要求更改系统的架构。我们可以在保证网站稳定的前提下，做一些小规模独立变更，尽可能优化网站的速度。

7) 实践出真知，切记，做技术最怕的是人云亦云。现在 IT 技术发展很快，有许多新技术正逐渐被开发出来，有些在测试环境下可成功通过的技术未必适用于生产环境，在使用一项新技术时，我们应该抱着稳定大于一切的宗旨进行实验和尝试，还应该结合网站的实际情况，因地制宜。

在企业中，无所事事的系统管理员未必是不负责任的员工，相反，很有可能他们是优秀的人才。当我因为工作而忙得体力透支时，我就会反思自己：到底是因为能力的原因不能胜任工作，还

是我自己的工作方法出了问题，抑或是公司的体制问题？不要因为一些人为因素影响了工作。很多时候，多思考，再工作，少做无用功，这样的工作方式才是正确的工作方法。

10.6 给 Linux/Unix 新人的建议

我有幸担任过一段时间的 RHCE 教学工作，我发现许多学员对为什么学习 Linux 感到很迷惘，更别谈什么职业规划了。我在工作和学习中，也接触了不少学习 Linux 的朋友，但发现他们在学习的过程中难免会走进误区，白白浪费了不少精力。本节希望通过说明 Linux/Unix 的从业现状，来帮助大家走出学习的误区。

通过多年跟踪学员的情况我们发现，学员在同等条件下学习 Linux 后，应聘开发职位所获得的薪水要比系统应用职位上升得更为迅速，更容易突破 5000 元/月、8000 元/月甚至 10 000 元/月。但是这个方向难度会更大，同时也会更枯燥些，需要毅力和身体，更需要兴趣。如何入门及如何规划自己的 Linux 学习之路一直是困扰 Linux 爱好者的一大难题，实际上，Linux 的入门与学习并不难，只是由于长期使用其他操作系统的原因，以及没有正确的引导指向，使得一个本身不是难题的问题变成了难题。

这里我想给读者提一个问题：企业为什么要用 Linux 或 Unix？也许会有许多读者会说 Linux/Unix 更好，更稳定。其实也并不全是这个原因。企业为什么要用它们，其实是本着节约成本和创造价值的目的。服务器操作系统的软件投入和服务投入是一笔相当大的经费，这也是开源免费的 Centos 和 FreeBSD 越来越受欢迎的原因之一。再说一下硬件的负载均衡器，比如大家最熟悉的 F5，性能一般的要十几万元人民币，如果要想加新功能、新模块，价格也会随之水涨船高。但其实许多企业所需要的 Linux 集群环境的并发并不是太大，完全可以用开源免费的软件来代替，例如 LVS、HAProxy，而这些软件一般都是部署在 Linux 上的，这也是近年来 Linux 越来越受欢迎的原因之一。

另外，学习 Linux 的一个误区是只局限于学习 Linux 本身，从 Linux 的操作到 Linux 的内核都学，也不管自己到底要干什么。千万不能这样做，Linux 只是操作系统，重要的是其上的应用。系统是拿来用的，学习使用 Linux 的目的是创造效益，如果不是仅仅为了制作 Linux 系统，不是加入了 RedHat 这样的专业 Linux 发布版厂商中，那么你在学习了 Linux 基础知识后就跟掌握了 Windows 没什么两样。很显然这是不够的，你还需要更进一步的学习，比如：选择 Linux 上的应用，或者 Linux 上的软件开发，当然还有数据库方向。下面我就这 3 个方向的发展详细说明一下。

(1) Linux 的系统、网络、服务、集群等方向如下：

- Web 应用服务器，如 sina、百度等大型网站。
- Mail 应用服务器，如 163 或外企 mail 系统等。
- 中间件或 J2EE 服务器，如为 JBOSS Weblogic 做平台。
- 网络应用等。

(2) 嵌入式开发、Linux/Unix 应用系统开发、Linux 内核驱动开发，主要有以下几类：

- Linux 下的 C/C++ 系统程序开发。
- Linux 平台 Java 体系开发和 PHP 开发。
- Linux 下的图形界面开发。
- Linux 底层内核/驱动开发。
- 嵌入式 Linux 开发等。

(3) Linux 下的数据库, 如 MySQL 数据库、Oracle 数据库和 Windows 下的 SQL Server 2008 等。数据库的重要性我就不重复了, 而 DBA 在企业中的作用称得上举足轻重, 在技术类岗位的薪资待遇方面, 总监之下就是数据库 DBA 了。

以上是 Linux/Unix 发展的 3 个方向, 大家可以在平时的工作和学习中有所偏重, 3 个专业精通一个即可。注意我这里说的是精通, 而非仅仅熟悉的程度, 意思是你的技术放在企业里要马上就能用, 而且你掌握的技能 and 技巧要能让企业的生产环境稳定无故障地运行。

如果按照地域划分, Linux/Unix 系统运维方面的工作适合在北京、上海、深圳和广州等地发展, Linux 下的开发比较适合在杭州和南京等地区发展。相对而言, 开发人员的待遇普遍要高些, 但这真的不需要羡慕, 别人也付出了常人难以想象的努力。

我这里想说的是: 如果是初学 Linux/Unix 的读者, 建议以 Centos 为主, 在找工作时尽量不要考虑待遇的问题, 在工作实践中, 尽量熟悉 SHELL 和一些基础的网络应用, 另外, 内网开发服务器的配置环境一定要熟悉。如果有机会外出做项目, 一定要兢兢业业, 因为这个是成长最快最好的时候。另外, 无论是在学习还是施工期间, 有关环境配置一定要形成文档, 而且要尽量写得详细和完善些, 这样无论是对自身技术的提高还是跟同事进行工作交接, 都会非常有帮助。如果没有对外项目的机会, 可以写技术型博客, 详细记录自己的学习过程, 然后跟一些爱好者互相交流学习。如果想进一步提高自己的水平, 建议多上一些跟 Linux/Unix 有关的论坛, 多跟技术大牛们交流探讨, 这样会非常快地提高自己的水平。

如果是资深的 Linux/Unix 系统管理员, 建议脱离系统级别, 向系统架构师的发向发展。系统架构是件艺术活, 有时系统性能好坏就要看网站架构师的能力了。通常情况下, 此职位由公司的技术总监担任, 它要求技术总监对系统、程序、网络及数据库都有相当的了解, 希望本节内容对广大的 Linux 新人有所帮助!

10.7 系统管理员之企业生存守则

我是 2005 年参加工作的, 期间做过红帽的 RHCE 培训讲师及项目实施工程师, 目前在一家外企做 Linux/Unix 系统管理员的工作。我感觉系统管理员在企业中生存还是相当需要技巧的, 这里有一些心得, 跟大家交流一下。

1) 不要太封闭自己, 友好地与非本部门的同事相处也很重要。俗话说得好: 先做人, 再做事。良好的人际关系是你成功的关键条件之一, 也是愉悦工作的基本条件之一。千万不要以为技术第一, 其实在技术人成功的条件之中, 技术未必是排在第一位的。在公司的人事架构中, 技术类岗位往往是排在中下位的, 所以我觉得仅仅跟本部门的技术类同事打交道是不够的, 你应该多跟其他部门的同事 (如行政部、人事部等部门的同事) 多接触一下, 多了解公司的企业文化和内部规定, 以及人事架构, 这样对自己的成长也是有帮助的。我以前在公司上班时, 往往过了 3 个多月都不知道自己公司的董事长和总经理长什么样, 其实这样不好, 万一哪天在他们心里留下了一个目空一切的印象, 很影响仕途哦! 尽量在不影响公司的内部规定的前提下, 帮助能帮助的人, 多跟其他部门的同事聊聊, 多沟通, 这样就算你是在一个新的公司里, 也能够很快适应公司的工作氛围, 进入工作状态。

2) 正确处理跟本部门同事的关系。有句老话: 不要跟同事做朋友。很不幸, 这句话在系统管理员身上并不适用。如果是一个大型的公司, 比如超过 500 名员工的分公司, IT 部门划为开发组、

系统组、网络组（包括网络安全），有许多工作要求协同工作，仅凭一人之力是不能完成工作的。所以这时候你需要花的大量时间跟你的同事（如 PHP 开发人员或网工们）沟通，让他们明白或了解你的需求。特别是一些开发环境的部署，因为最后的测试使用者正是你的开发同事们。打比方说，我在某公司做系统管理时，我们的测试环境是 Nginx0.8.46 + PHP5.2.14 (FastCGI)，而 PHP 开发人员当时使用的是 Zend Framework，他们对 Nginx 的要求就是要有统一的入口，这就需要在 Nginx 上写相关的匹配的正则分发表达式了。个人认为，如果同事们在私底也能聊得来的话，不妨也可以作为朋友交往。如果确实存在工作利益上的冲突，完全可以以商量的口气来协调工作。切忌的两件事是：不要以技术压人；另外，也不要以老员工的身份欺负技术新人。系统管理的工作其实就是搭积木，只要你愿意花时间，基本上没什么难度。而网络及网络安全这块，我跟同事沟通得就更多了，比如要将内网的某台服务器作 DMZ 映射；如果新网站要推行了，还要跟负责网络安全的同事们协同工作，看有哪些安全漏洞，或者查看防火墙的安全性能及服务器的最大压力承载情况等。建议平时也学习一些系统之外的知识，闲时也可以跟同事们多聊些技术以外的话题，比如手机、游戏或别的什么。周末方便的话，尽量参加公司或同事们的聚会。不要冷冰冰地做人，一张笑脸比什么都重要。在处理与本部门同事的关系时，我的做法是：能做同事就做同事，能做朋友就尽量做朋友，毕竟多一个朋友多一条路。所以，以前公司的同事们，只要聊得来的，我基本上都会保持联系，平时或周末也会跟他们聚一下餐，聊聊天，既减轻压力，又相互了解对方公司的一些趣事，何乐而不为呢？

3) 遇见领导要服软。这里也有两个原则：一是原则性的问题要服从上级领导的管理；二是千万不要越级报告，无论是国营还是外企。建议刚参加工作的读者仔细阅读一下《杜拉拉升职记》，里面许多故事都是挺真实的，特别是越级报告这个问题，短期来看，你可能会取得局部的胜利，但从长远来看，你绝对是最大的输家，因为没有领导会喜欢一个越级报告的下属，哪怕你的能力再高也是一样。我的第一份工作是在某大型国营企业里做网管，主要是负责 Windows Server 2000 服务器及 DB2 数据库，当时自以为技术牛（在公司里技术确实也算是第一），再加上很快就被提为了 IT 部门的 Leader，所以很有些飘飘然，在领导面前表现得不够尊敬，结果很快发现仕途不顺，一直只能是 Leader。其实如果当时明白这一真理的话，我现在估计也是朝技术 + 管理这一目标一直走下去的，也就没必要转实施工程师了。我当时就很迷惑：为什么许多没有能力的人都当了 Manager，而我还只是一个 Leader？其实当时就是没处理好与领导的关系，可能是太年轻和性格比较外向的缘故。告诉大家这一经历，也是希望大家引以为戒，特别是希望做管理的朋友更要注意了。有时候，你的上级能力可能没你强，也有可能不懂系统这块，你就更有必要耐心地向他解释，为什么不能这样做，这样做会带来怎样的恶果。千万不要消极对抗，尤其不要发生正面冲突。其实关于这个问题，大家到了一定年龄层次就会明白了。不过，我觉得提前明白还是有好处的，这样可以少走许多弯路，至少杜拉拉们明白这点。

4) 明确你的发展定位也是很重要的。作为一个系统管理员，即 System Admin，你要明白你的发展定位，到底是做技术 + 技术，还是做技术 + 管理，还是做技术 + 销售。这决定了你在相关方向投入的时间和精力。相信技术 + 管理大家都明白，技术 + 技术是一个怎样的定位呢？许多公司都应该有这样的岗位，即高级开发工程师，此岗位的薪资跟 Manager 持平，但不涉及人事管理。比较大的公司，也有高级系统管理员一职，我在北京的岗位也跟这个类似，系统总监不属于此岗位，它属于系统 + 管理。技术 + 销售就比较好理解了，即售前工程师和售后工程师，它们的技术含量跟系统

管理（系统集成）比较起来就比较低了，特别是售前，这是我比较喜欢的职位之一，如果是做过项目实施工程师工作的读者可以考虑一下，特别是大公司的售前，福利待遇相当不错，刚毕业的朋友可以利用这个机会过渡一下。每个系统管理员都应该明确自己的发展定位，做到有的放矢，合理分配自己的精力和时间。另外，说个题外话，英语对系统管理员很重要，因为许多新产品和新技术，基本上都是从国外引进的，要想熟练地掌握及应用它，英文是必不可少的基本功之一。外企就更不用说了，我们向国外的上级 Leader 报告时，其正式文档均要求是英文的。搞技术的人都容易忽略的另一条就是口才，其实这也是很重要的。尤其是作为售前工程师，你总不可能在向你的客户推销你公司的产品方案时就直接让客户阅读文档吧，或者就直接告诉他们“这个好，这个确实好”吧。不说别的，大家面试时，一个技术人员如果有一个好口才，其发展方向也可以是多方面的，至少你还可以成为职业讲师，让更多的人学到你的专业知识。

5) 系统管理员一定要明确自己的企业定位。老板们现在越来越喜欢 Linux/Unix 的原因未必就是你所想象的那样，比如 Linux/Unix 是多么的有效率。据我所知，很大部分原因是因为 Linux/Unix 免费，而且它下面的许多软件也都是免费开源的，比如 Apache、LVS、Nginx、Squid 及 Postfix，当然还有 bind9 和 XEN，还有 iptables，都比较强大。我就职的公司，至少有 60% 以上是用 iptables 作为 NAT 路由器，而且其效果不错。作为系统管理员，你要明白你在企业中的真正作用：并不是你的位置有多重要，而是你能将技术转化为生产力，为公司能节约多少成本。所以千万不要以为公司缺了你不转了，一定要抱着平常心的态度去工作和生活。我现在认识的一些技术高手们，基本上都很谦虚和平民化，这也值得我们学习。另外，平时要注意学习，现在新技术层出不穷，能力不是天生的，这个需要后天培养。你还可以通过博客等形式发布自己的工作或学习心得，或是率先掌握一门新技术，并率先向社会推广这门新技术。分享是一门艺术。在分享的同时，一定会伴随着理解、应用、总结、提高、表达甚至推广方面的提高，这对个人的技术提高和社会影响力的建立有着非常的意义，这也是我目前努力的方向和目标之一。

6) 工作一定要有效率，能简单就简单。其实作为系统管理员，许多工作都是重复性的，特别是一些维护和备份工作，这个时候你完全可以编写 SHELL 脚本，加到 Crontab 计划任务里，代替你在某时间段执行这些操作。Windows 下的批处理也是相当不错的，许多用图标的操作也可以简化。在你将工作都理顺后，你会发现你的工作原来就是如此简单。你完全可以将你的时间用于其他方面的学习，比如数据库和程序开发等，这也是我一直强调脚本重要性并特地单独将其作为一节来讲的原因之一。做一个优秀而“懒惰”的系统管理员，优秀的管理员绝对是一个“懒惰”的家伙，如果你作为系统管理员，每天都要加班加点地工作的话，这个时候不妨反思一下。

7) 系统管理员要明白自己在公司的作用。作为系统管理员，一般都会职守公司的 Exchange 邮件服务器，当然还有许多机密的文件，这时候一定要做好保密工作，不该说的话和不该做的事都不要做，尤其是涉及薪水方面的敏感话题时。在公司内部，透露和打探公司的薪资都是职场大忌。另外，随便透露公司的资料及敏感信息、上班间接私活等都不要做，这都是些犯忌讳的事情。还有一件事让人比较头疼，那就是每个公司，无论大小，都会有一些小团体斗争，这个时候该怎么办呢？我一般的做法是，绝不拉帮结派，尽量保持中立，做事要对得起自己的良心。如果确实做不到，那就考虑离职吧。毕竟做人是一辈子的，做事是一时的。下次工作时，记得找一个工作环境单纯点的公司。其实如果遇见这些事也未必是坏事，至少下次遇到时就不会惊惶失措了。我在公司里尽量做到以下两点：注意保密，保持中立。

8) 下面要说的是与身体健康相关的事情了。有时候,服务器迁徙的活儿还是比较重的,1U、2U的服务器还好说,4U的就比较重了。我的身体比较单薄,独自一人勉强能应付1U的服务器,其他的就不行了。另外就是夜班值守的问题,这就比较头疼了。我一般就是白天注意多休息一下,晚班的时候会将手机邮件开通,音量调到最大,下半夜时能睡会儿就睡一会儿。别的网站崩溃了不要紧,如果是电子商务型和广告类型的,那就是钱了,这也是老板最紧张的。所以系统管理员也要注意锻炼身体,可以办一张健身卡,周末去锻炼一下身体,平时能走路的话就不要打车了。另外,要注意调整心理方面的压力,因为我们的平均故障处理时间不能超过5分钟,所以上班时的压力还是很大的,我有段时间还因此而掉头发。所以,我现在喜欢讲讲笑话,跟同事们聊聊天,保持轻松的心情和良好感觉。有时坐多了,会有些颈椎之类的小毛病,平时也要注意调整。身体是工作的本钱,如果你的工作直接影响到你的健康的话,建议还是换一份工作吧。没有什么事情比健康更重要。有了健康,才有将来,这句话对我们也适用。

说了这么多,都是自己这么多年来的一些经验和心得,在职场上生存和发展不容易,希望能给有志于向系统集成方向发展的朋友一些帮助,也给目前正在从事系统管理员一职的朋友们一些帮助吧。

10.8 小结

在本章里根据我自己在企业中的工作心得,详细说明了如何学好Linux,以及Linux系统管理员在企业中如何工作,在企业中如何生存及发展,最后也说明了Linux学习的几个方向。希望大家可以通过此章内容确定自己的学习目标,如果能做到有目的地学习和工作,那么我撰写此书的目的也算是达成了。

附录 A Xmanager 3.0 企业版实用技巧集锦

Xmanager 3.0 是一个运行于 MS Windows 平台上的高性能的 X Window 服务器，利用它你可以在你的本地 PC 上同时运行 Linux/Unix 和 Windows 图形应用程序。它使得用户能轻松和安全地从 Windows PC 上访问 Linux/Unix 主机，它自带的 Xftp 让你能安全地在 Linux/Unix 和 Windows PC 之间传输文件。Xmanager 3.0 企业版是收费的，大家也可以先用免费而强大的 Xshell3.0。Xmanager 3.0 企业版安装完毕后有相当多的组件，如图 A-1 所示。我用得比较多的是 Xshell3.0、Xbrowser、Xftp，大家可以尝试使用一下 Xlpd，它是一个用于 MS Windows 平台的 LPD（行式打印机虚拟后台程序）应用程序。



图 A-1 Xmanager 3.0 安装成功后的组件工具

安装了 Xlpd 后，你的带有打印机的本地 PC 就成为了一个打印服务器，来自不同远程系统的打印任务都能在网络环境中被请求和处理。

Xmanager 3.0 的特点如下：

- 跟 PuTTY 不同，它只需要一个程序窗口就可以同时控制成百台 Linux/Unix 机器，尤其适用于超过 500 台 Linux/Unix 主机的分布式环境。
- Xshell 的乱码处理情况比 PuTTY 好得多。
- Xbrowser 能很好的从 Windows 上控制 Linux 桌面，尤其是需要操作 Linux 下的 Oracle 数据库时。
- Xsftp 可以很方便和安全地上传和下载 Linux/Unix 服务器的东西，尤其是在 Windows XP 下。它支持 FTP 和 SFTP，有了它我基本上就不用 WinSCP 和 FileZilla Client 了。
- 支持同时操控 N 台 Linux 主机，适合中等规模的分布式环境，部署大规模的分布式环境时我们可以考虑用 Puppet。
- 它相当于 PieTTY + WinSCP + 远程桌面的组合，这套工具用得熟的话，你的机器可以少装许多软件，还可以达到高效工作的目的。

1) 在 Xshell3.0 下如何解决远程乱码的问题（此方法也能解决 PuTTY 的乱码问题）。

我的服务器一般是英文最小化安装，如果有中文需求，可先安装中文语言包，方法如下：


```
yum -y groupinstall chinese*
```

如果 Xshell3.0 或 PuTTY 出现了乱码问题，我们可以用如下方法解决：打开 PuTTY 主程序，选择 window→Appearance→Font settings→Change...，然后选择 Fixedsys 字体，字符集选择 CHINESE_GB2312，在 window→Appearance→Translation→Received data assumed to be in which character set 中，把 Use font encoding 改为 UTF-8，一般这样就行了。如果不行，接着往下看（以下办法适用于中英文版本）。

我们接着需要进行如下步骤（建议以下两个步骤均操作）：

a) console 终端乱码。

在/etc/profile 文件的最后一行添加如下内容：

```
export LC_ALL="zh_CN.GB18030"
```

b) X-window 终端乱码。

在/etc/sysconfig/i18n 文件的最后一行添加如下内容：

```
export LC_ALL="zh_CN.GB18030"
```

按以上步骤操作以后，就不会出现连执行 setup 都出现乱码的问题了。其实 Putty 出现乱码不外乎就是编码及字符集的原因，我在 RHEL4 及 Centos5.2 | 5.3 | 5.4 | 5.5 上均能通过以上步骤来实现，解决乱码问题后的截图如图 A-2 和图 A-3 所示。

2) 我们可以利用 Xshell3.0 连接 N 台远程 Linux 或 Unix 主机。

关于 Xshell3.0，用 Linux 的朋友们相当熟悉，如果是一两台服务器需要连接的话，用 PuTTY 相当快捷和方便，但要是维护 CDN 节点的成百台 Centos 时该怎么办呢？我们内网的开发服务器及 Nagios 监控加起来差不多有 20 多台，如果都用 PuTTY 来远程的话，就要开 20 个远程，就算加上 Screen 也要开许多 PuTTY，这就让人比较郁闷了。不过，Xshell3.0 可轻松解决这个问题，大家可以看一下图 A-4，那是用 Xshell3.0 连接 4 台内网开发机器的截图。

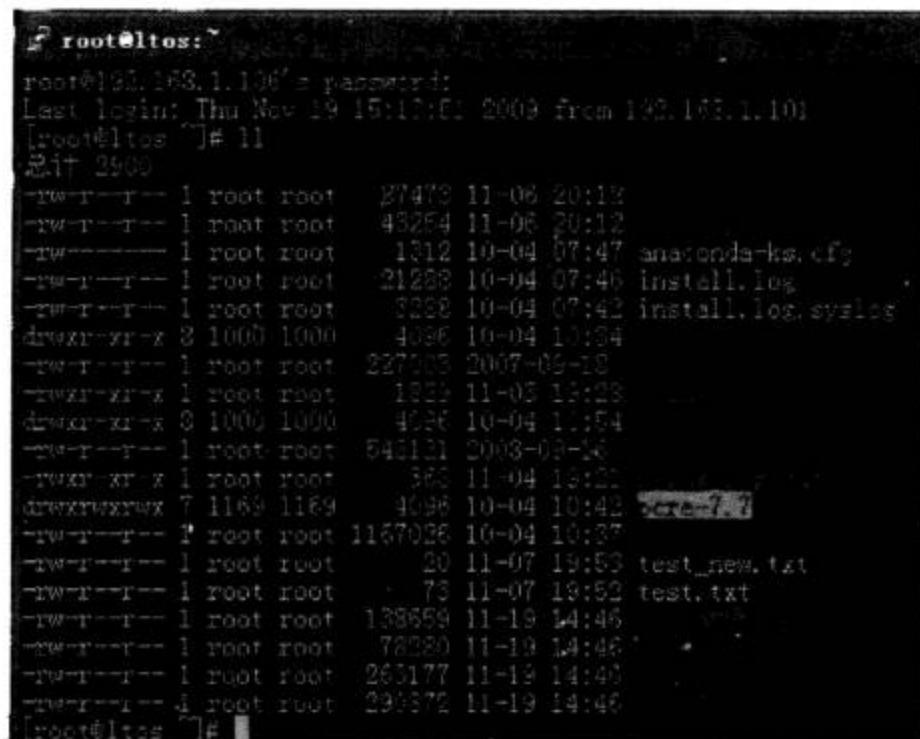


图 A-2 PuTTY 的中文正常显示截图

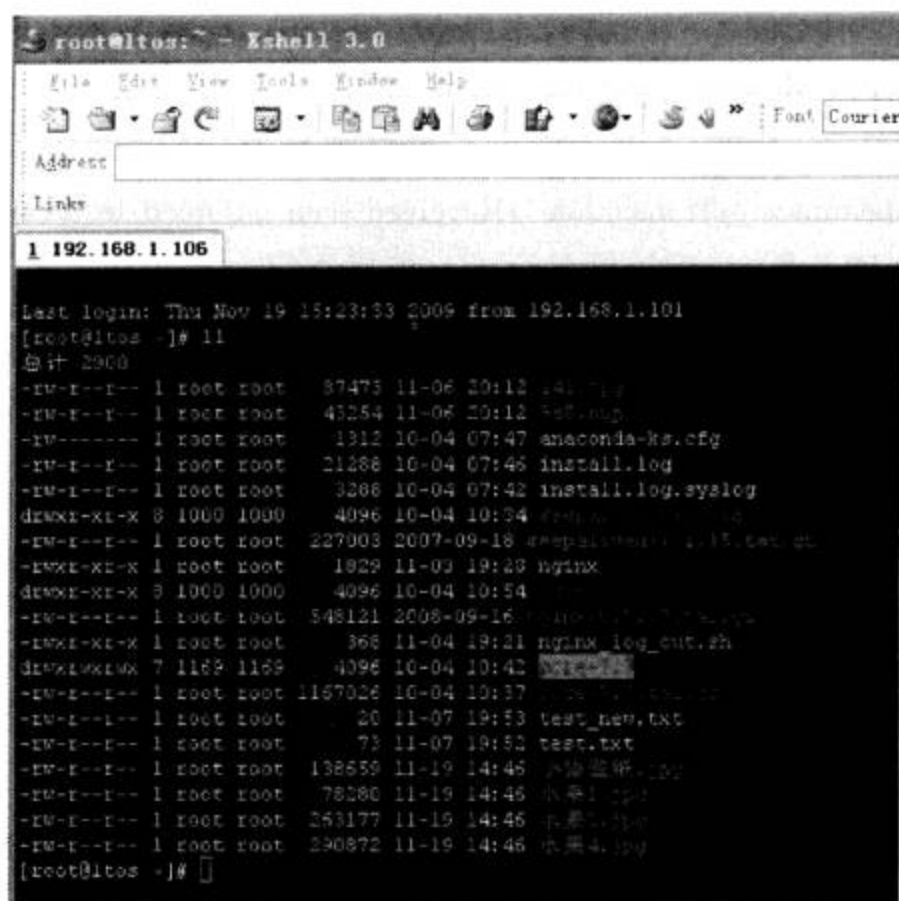


图 A-3 Xshell3.0 中文正常显示截图

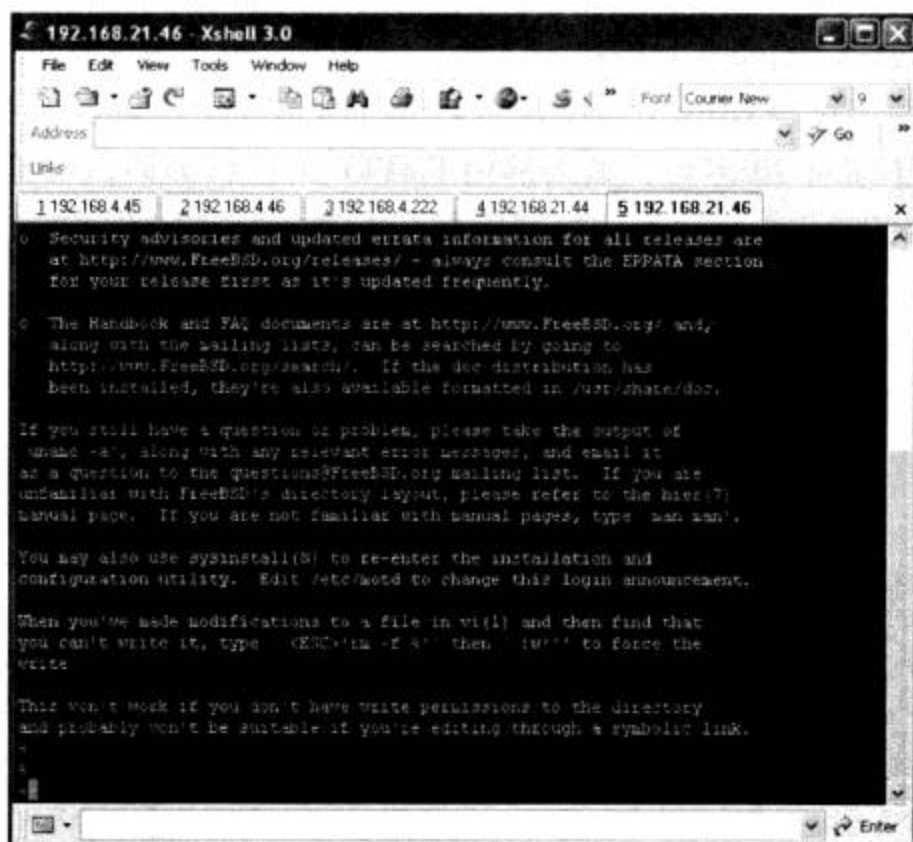


图 A-4 Xshell3.0 连接 4 台 Linux/Unix 机器的截图

如果要上传文件，大家可以直接点中要上传的文件拖到 Xshell 里面，很方便。大家注意观察就会发现，这里用的还是 rz 协议，如图 A-5 所示。如果要下载呢，我们不是还有 X-FTP 工具吗？

如果要从自己的博客或别人的博客上复制粘贴代码，先选中，然后直接复制和粘贴，跟 Windows XP 的操作一样。



图 A-5 在 Xshell3.0 直接拖动上传文件

- 3) 用 Xbrowser 来连接 Linux 的 X Windows 桌面。以 Centos5.x 来说明一下，详细配置过程如下：
- a) 我们先设置好相应的 Centos 系统的相关配置，建议在 Centos5.x 系统上安装 X Windows。

For XDMCP connection to RedHat 5 EL

①XDM Configuration

②Change runlevel to 5

Open /etc/inittab and set the initial runlevel to 5 as following:

id:5:initdefault:

③Enable XDMCP

For GDM:

a, Open /etc/gdm/custom.conf and set the Enable entry to 1 in the [xdmcp] section as following.

[xdmcp]

Enable = 1

#此项没有就添加 Enable = 1

b, Firewall (TCP/UDP Ports) Configuration

Open UDP port 177 from the PC to the remote host direction.

Open incoming TCP ports 6000 ~ 6010 from the remote host to your PC.

c, Reboot the remote host and start Xmanager

init 3; init 5

重启计算机后，我们要检查一下 Centos5.x 系统的 177 端口打开没有，命令如下：

lsof -i:177

b) 如果我们这里用 root 远程登录 X Windows，要做如下变动。

在 /etc/gdm/custom.conf 文件里增加如下内容：

vim /etc/gdm/custom.conf

[security]

AllowRemoteRoot = true

在 /etc/securetty 文件的结尾处增加以下内容：

vim /etc/securetty

pts/0

pts/1

```
pts/2
pts/3
pts/4
```

编辑 `vim /etc/pam.d/login` 文件，所做的变动如下所示：

注释第一行：

```
#auth [user_unknown=ignore success=ok ignore=ignore default=bad] pam_securetty.so
```

我们在 `/etc/pam.d/remote` 文件里注释第一行，如下所示：

```
#auth      required      pam_securetty.s
```

c) 如何用 Xbrowser 建立一个远程的连接呢？大家可以跟着我一步一步地操作，如图 A-6 至图 A-9 所示。

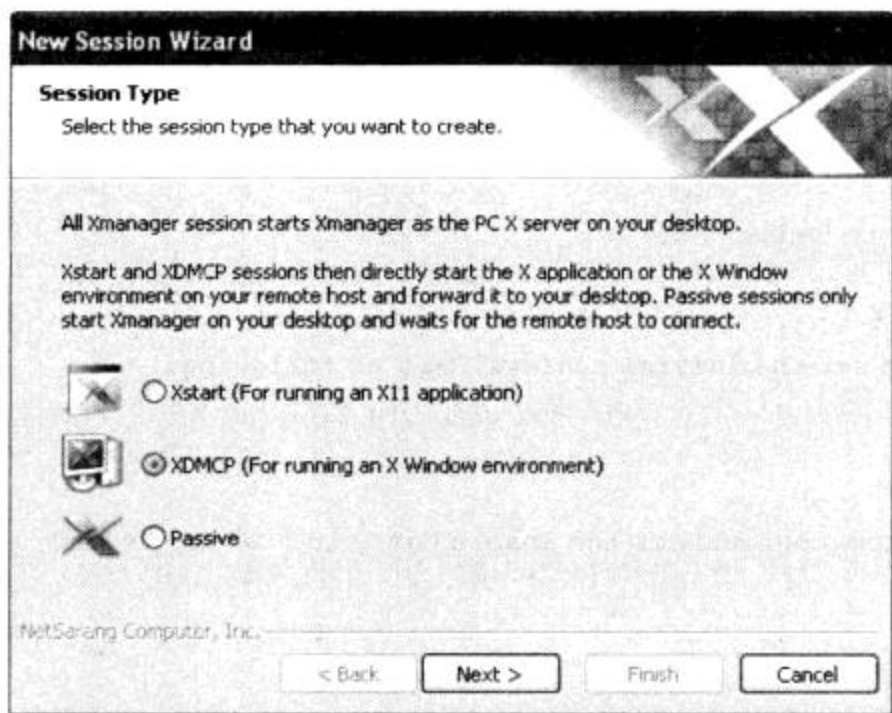


图 A-6 打开 Xbrowser 工作界面

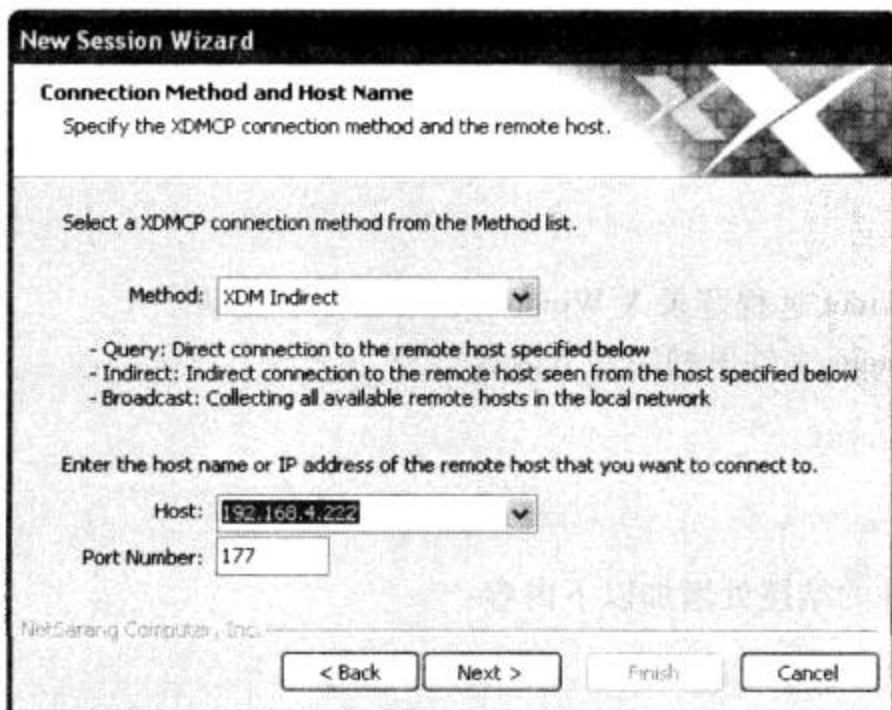


图 A-7 选择我们要连接的远程主机的 IP 及端口号

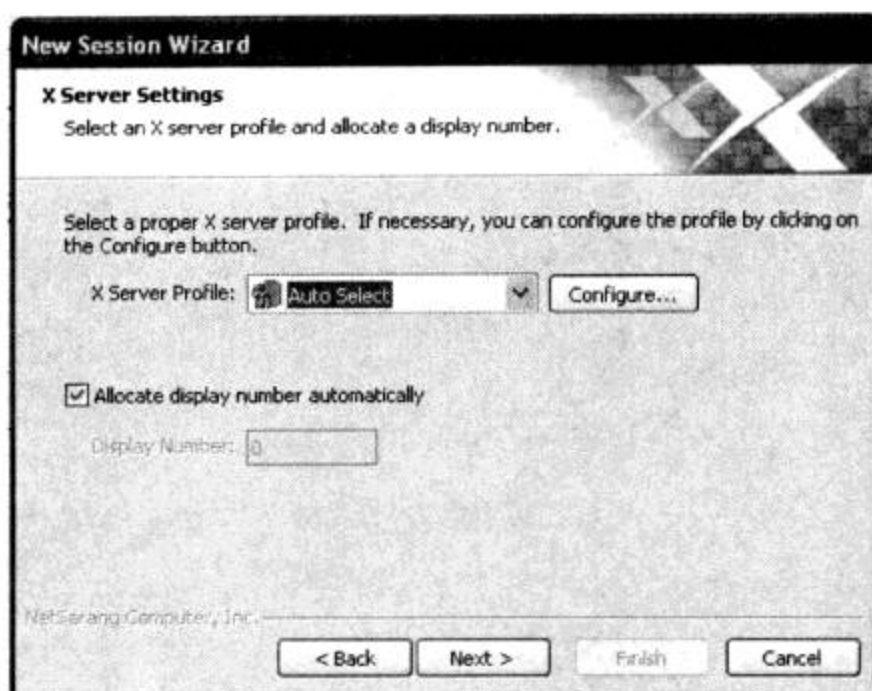


图 A-8 X Server 配置文件选择默认

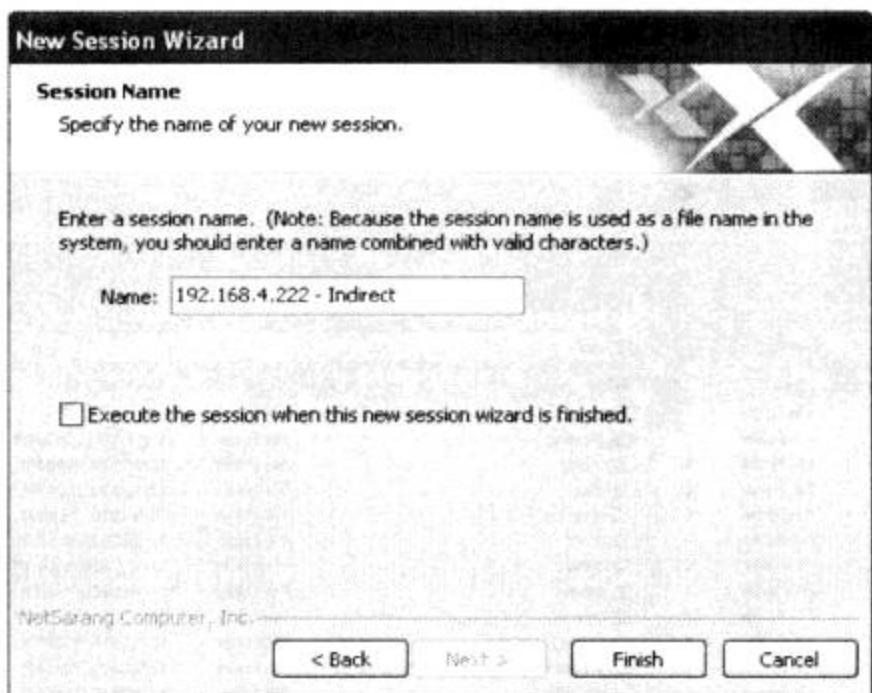


图 A-9 选择新建连接的名称

整个配置过程结束后，我们就可以轻松地使用 Linux 的 X Windows 了。我在数百台 RHEL&Centos 机器上均尝试过以上服务器的设置和 Xbrowser，都能成功。效果截图如图 A-10 所示。

4) Xmanager3.0 企业版的另一个亮点，就是它的 Xftp，它支持在 Windows 和 Linux/Unix 之间互相上传和下载文件，它用的是安全的 22 端口，即 SFTP 协议。当然了，用它连接 FTP 服务器就更没有什么问题了。它的使用也极简单，跟 CuteFTP 和 Winscp 的操作界面一样，可傻瓜式地操作。我在以前的公司中推广 Xmanager3.0 企业版时，基本上只要让同事看看效果，他们就会用此工具了。Xftp 的工作界面如图 A-11 所示。

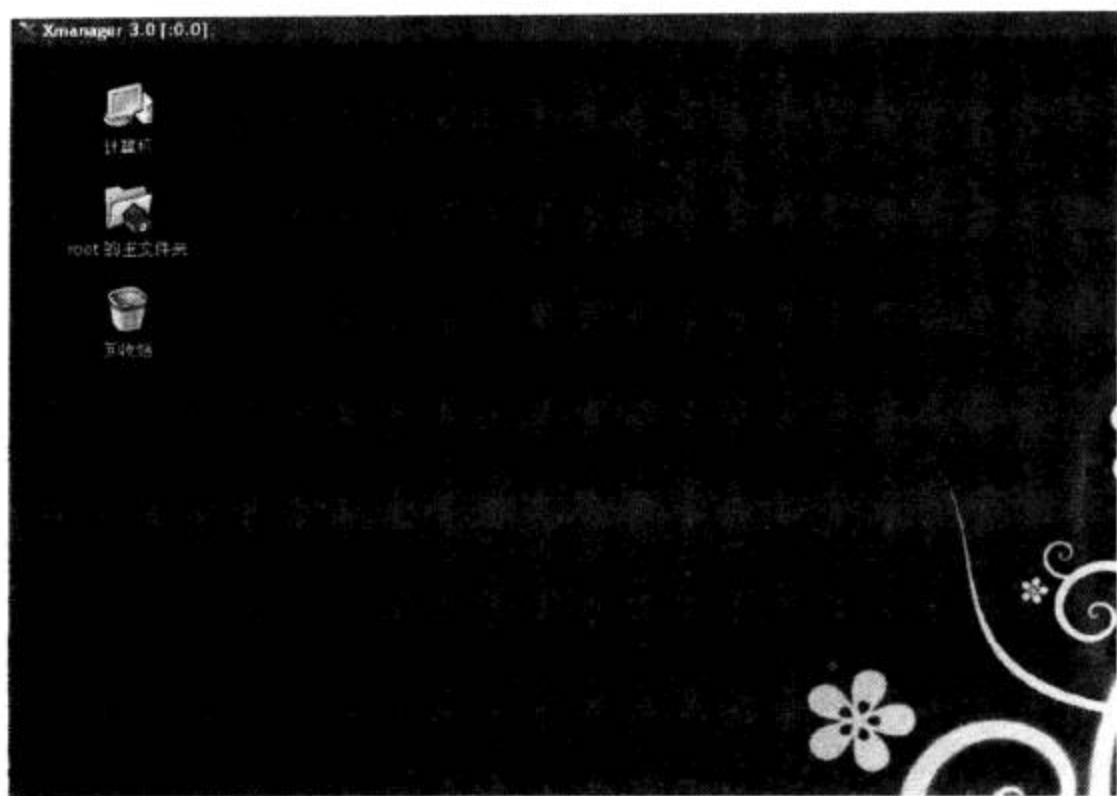


图 A-10 Xbrowser 远程连接 Linux 的 X Windows 截图

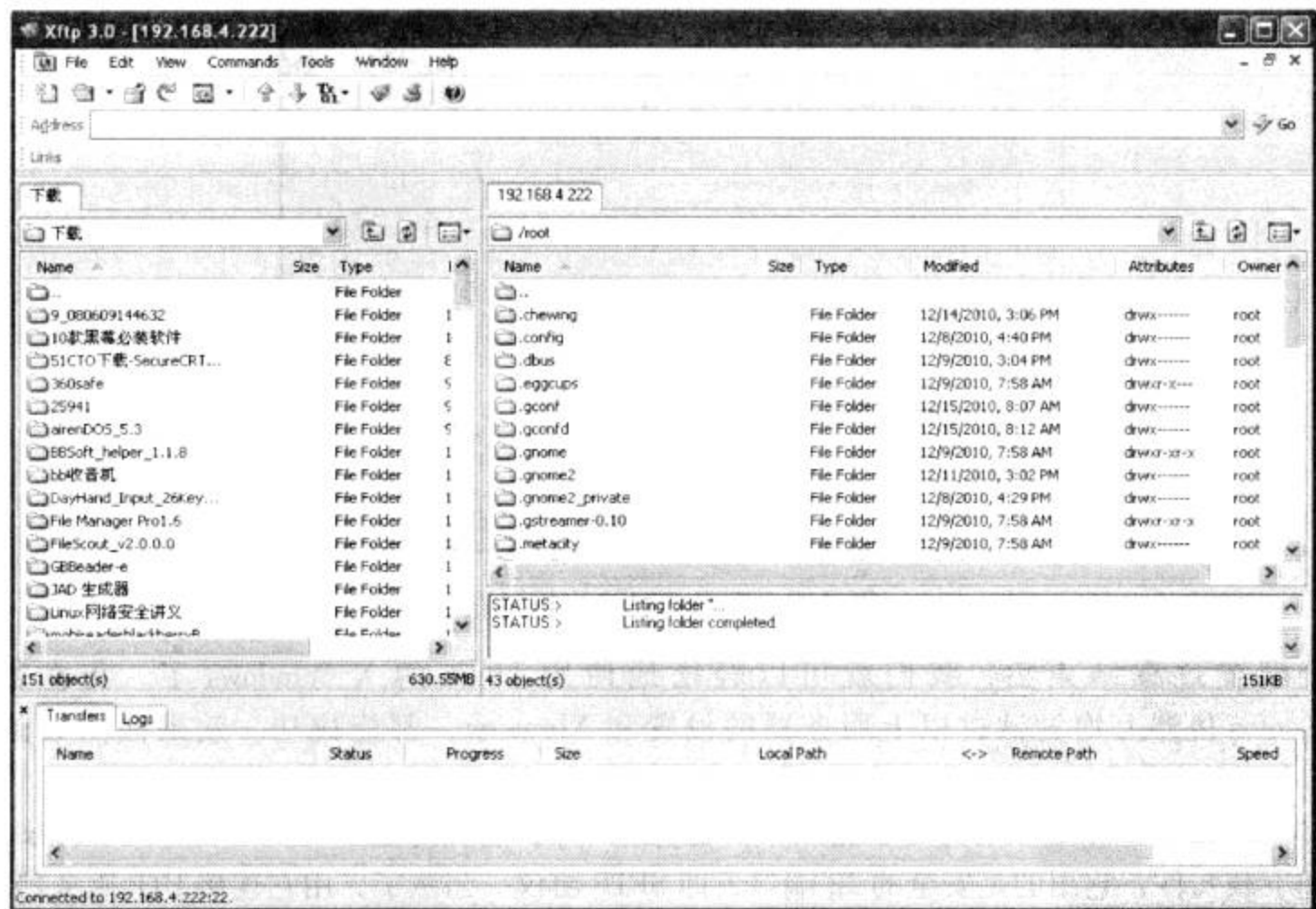


图 A-11 Xftp 工作界面图示

5) Xshell3.0 可以同时操控几台 Linux 主机，这个功能尤其适合 N 台 Web 主机或 Squid 一起部署，非常方便。我们输入命令时采取“To All Session”，就可以对此时连接的所有主机同时进行操作了，如图 A-12 所示。

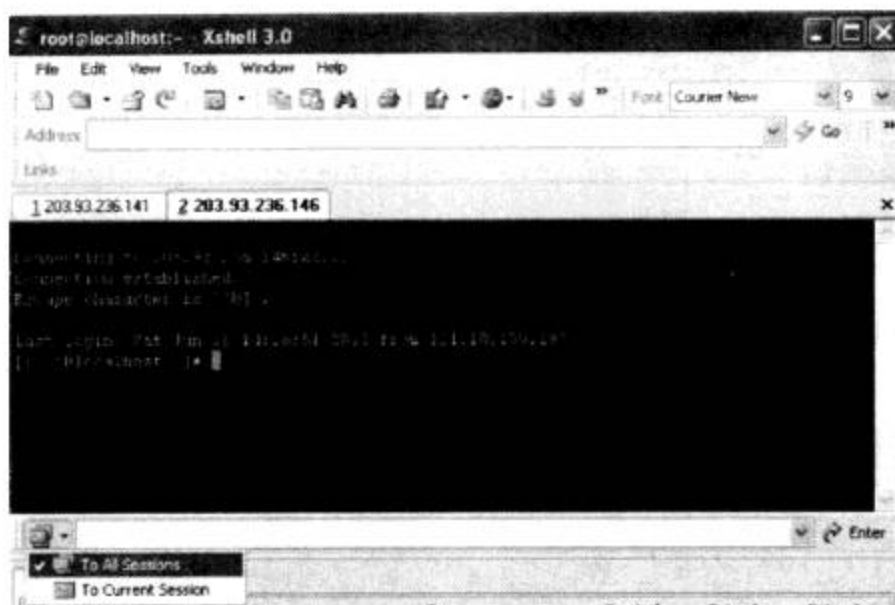


图 A-12 同时对 IP 为 203.93.236.141 和 203.93.236.146 的主机进行操作

大家采用 Xshell3.0 时应该还注意到了，无论我们在服务器上停留多久，都不会被服务器的 SSH 踢出来，而要是采用 PuTTY 和 PieTTY 就没这么幸运了。以前许多搞开发的同事都在向我抱怨，说用 PieTTY 时经常被服务器踢出来，其实用 Xshell3.0 就可以避免这种情况。大家可以看一下图 A-13，注意 Xshell3.0 里有一项“Keep Alive”，它的功能应该是每 60 秒就向服务器发一次包，告诉服务器“我还活着，不要踢我”，PieTTY 和 PuTTY 中也有相应配置，大家进行相应的改动即可。

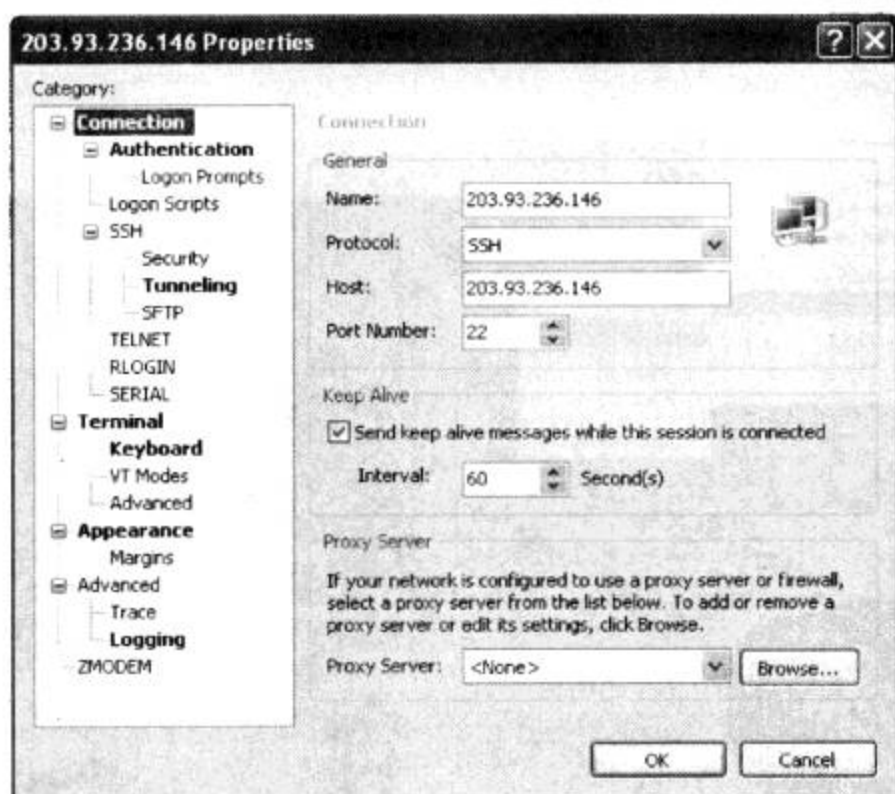


图 A-13 Xshell3.0 连接远程服务器的配置界面

6) Xshell3.0 的日志录像功能估计大家用得不多。下面将给大家演示一下。

步骤一，如图 A-14 所示。

它会提示要求你保存的日志名，然后输入获取日志的命令，假如是邮件服务器，可用 `tail -n 10000 /var/log/maillog`，即取最后的 10 000 条日志。

步骤二，如图 A-15 所示。

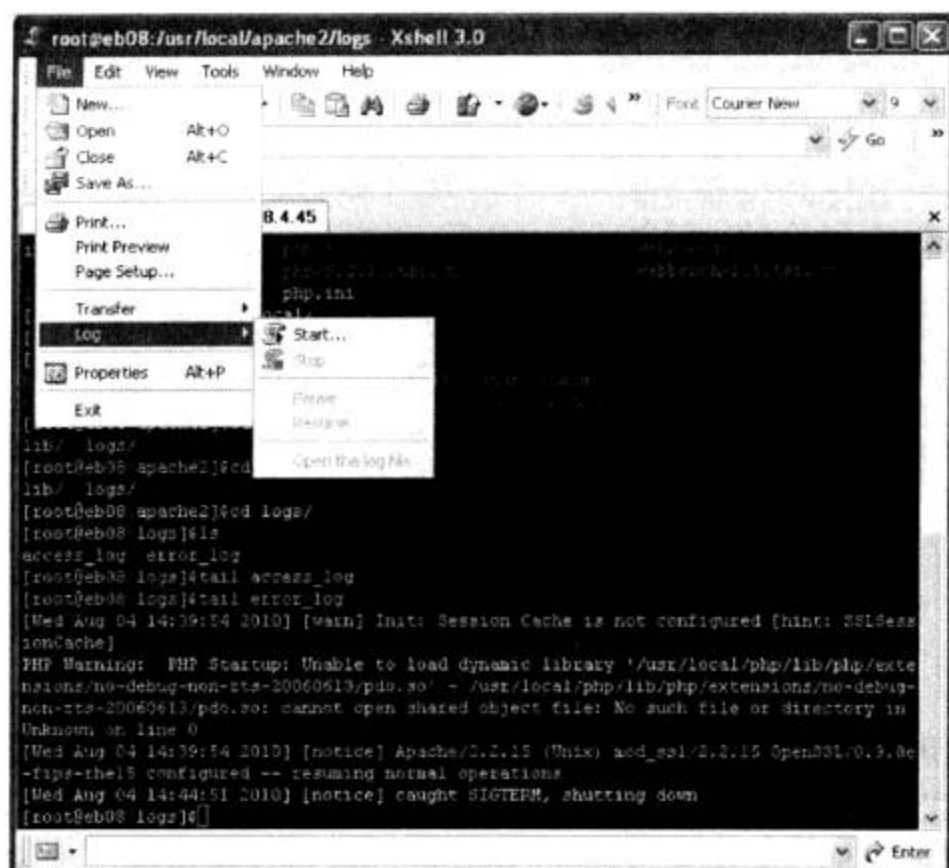


图 A-14 启动日志录像功能

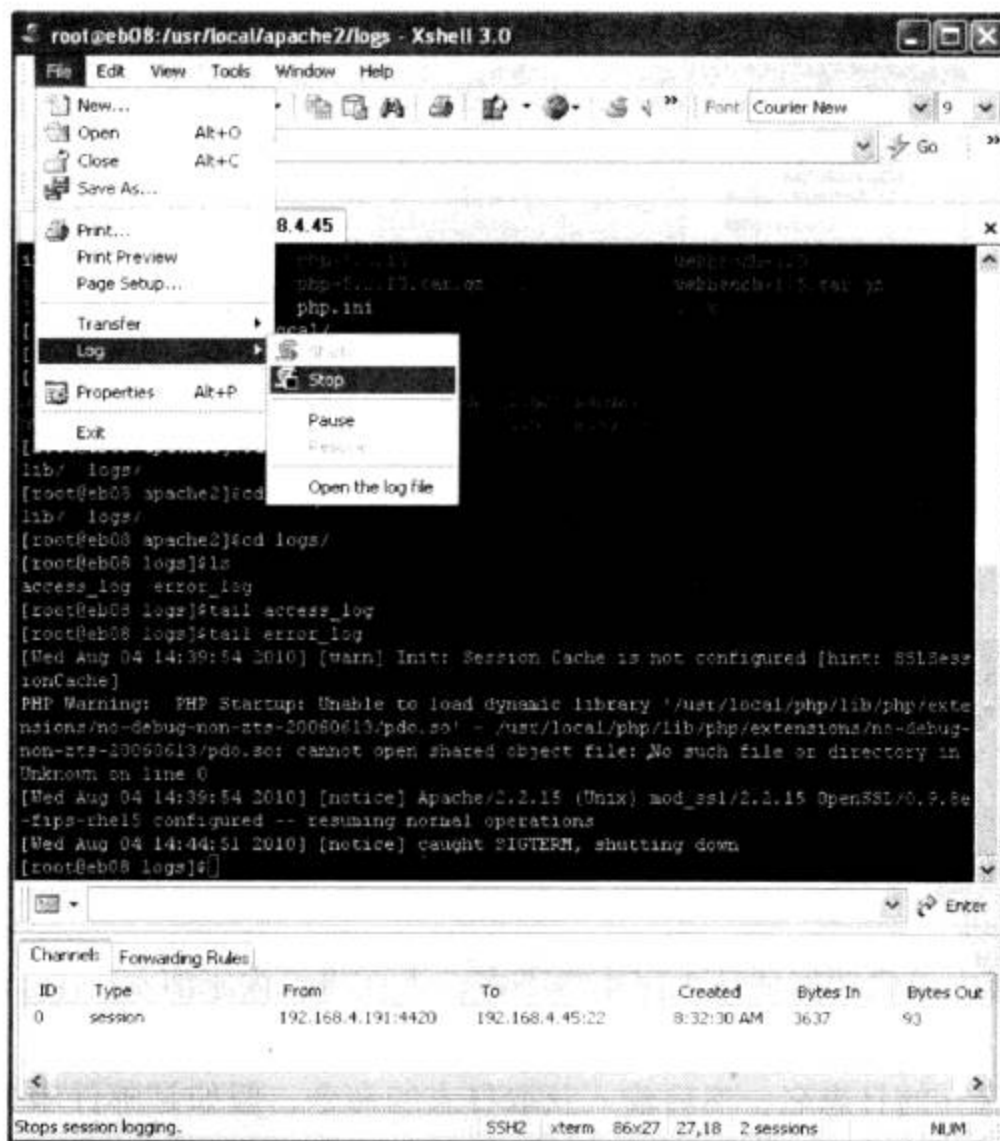


图 A-15 停止日志录像功能

这样日志就会自动保存到你的 Windows 上了，这时候我们可以用与 gVim 类似的工具来查看异常日志，既安全又方便。不喜欢在命令下用 Sed 和 Awk 的读者可以用此方法在 Windows 下用 Windows 的工具进行日志分析。

在远程操控 Linux/Unix 机器的工具中，PuTTY 虽然是老牌的 SSH 软件，但说老实话，它用起来太不方便了，我现在主要是用 PieTTY 来代替它。PieTTY + Xmanager3.0 基本能满足日常工作的需求了，至于 Windows 的远程桌面，我一般是用 mstsc 和 RemoteAdmin，有需求的话再加上 pcanywhere10.5。网上的相关文章众多，这里就不浪费篇幅了。希望大家熟练地使用 Xmanager3.0，让我们的工作更有效率。

附录 B 使用 Screen 管理远程会话

大家是不是经常需要使用 SSH 远程登录到 Linux 服务器上？是不是经常会为一些长时间运行的任务而头疼（比如系统备份、FTP 传输等）？通常情况下我们都是为每一个这样的任务开一个远程终端窗口，因为它们执行的时间太长了，必须要等它们执行完毕，在此期间可不能关掉窗口或断开连接，否则这个任务就会被杀掉，一切就半途而废了。

为什么我们不能关掉窗口或断开连接呢？是什么原因造成的？

元凶是 SIGHUP 信号。

让我们来看看为什么关掉窗口或断开连接会使得正在运行的程序死掉。

在 Linux/Unix 中，有这样几个概念：

- 进程组 (process group)：一个或多个进程的集合，每一个进程组有唯一一个进程组 ID，即进程组长进程的 ID。
- 会话期 (session)：一个或多个进程组的集合，有唯一一个会话期首进程 (session leader)。会话期 ID 为首进程的 ID。
- 会话期可以有一个单独的控制终端 (controlling terminal)。与控制终端连接的会话期首进程叫做控制进程 (controlling process)，当前与终端交互的进程称为前台进程组，其余进程组称为后台进程组。

根据 POSIX.1 定义可知：

- 挂断信号 (SIGHUP) 默认的动作是终止程序。
- 当终端接口检测到网络连接断开时，将挂断信号发送给控制进程 (会话期首进程)。
- 如果会话期首进程终止，则该信号发送到该会话期前台进程组中。
- 一个进程退出导致一个孤儿进程组产生时，如果任意一个孤儿进程组进程处于 STOP 状态，则发送 SIGHUP 和 SIGCONT 信号到该进程组中的所有进程中。

因此当网络断开或终端窗口关闭时，控制进程收到 SIGHUP 信号退出，这会导致该会话期内的其他进程也退出。要想在我们的网络断开或终端窗口关闭后，控制进程不受影响，可以采用 Screen 工具。

关于 Centos5.5 下的安装我们可以用 yum，即：

```
yum -y install screen
```

FreeBSD8.1 下面我们可以采取 pkg_add，即：

```
sudo pkg_add -r -v screen
```

进入和退出：

- screen -ls：列出当前有哪些 Screen 在运行。
- screen -S name：将 name 作为 Screen 的会话名启动一个新的 Screen，建议在实际工作中用名

字命令 Screen，这样更有效率。

- `screen -r name`: 回到会话名为 name 的 Screen。
 - `screen -d -r`: 这个命令可以把一个正在运行的 Screen 抢过来。
- 进入后对会话的管理（以下用法适用于开发人员）如下所示：
- `Ctrl + A C`: 创建一个新的 SHELL。
 - `Ctrl + A`: 在 SHELL 间切换。
 - `Ctrl + A N`: 切换到下一个 SHELL。
 - `Ctrl + A P`: 切换到上一个 SHELL。
 - `Ctrl + A 0..9`: 切换到窗口 0..9。
 - `Ctrl + A W`: 显示所有窗口列表。
 - `Ctrl + A D`: 退出 Screen 会话后可以通过 `Screen -r` 回来。

注意 `Ctrl + A C` 表示先按 `Ctrl + A`，再按 `C`。其他的类似。

作为系统管理员，下面的用法可能也是用得最多的。在远程操作 Linux 服务器时，会因为网络的意外故障造成会话中断，或者某些操作（如 update）会需要很长的时间，但是操作人员不能等到操作完毕后再下班回家，所以往往会回家重新进行更新，这个时候就可以借助 Screen 这个命令来重新建立起中断的会话了。而开发人员更喜欢看到的是程序的执行过程，Screen 也能很轻松地实现此功能，操作很简单。

首先使用 `screen -ls` 找到中断会话的 ID 号。

然后使用 `screen -r id` 就能看到你刚刚中断的操作了，如下所示：

```
[root@andrewy ~]# screen -ls
There is a screen on:
      3442.pts-0.zhuzhu      (Attached)
1 Socket in /var/run/screen/S-root.
[root@andrewy ~]# screen -r 3442
```

工作中我发现，用 `PieTTY + Screen` 来工作也是非常快速和有效率的，有兴趣的朋友可以尝试使用一下这两种工具。

附录 C 自动化部署管理工具 Puppet

系统管理员经常陷入一系列的重复任务中，比如升级软件包、管理配置文件、系统服务、cron 任务，以及添加新的配置、修复错误等。这些任务通常是重复低效的，解决这类任务最好是让它们自动化，于是出现了定制脚本。由于环境复杂，定制脚本和应用程序一再被重复开发，并且很难适合多种平台，灵活性和功能也很难保证，于是像 Puppet 这样的自动化配置管理工具便出现了。

在开源世界里，有很多配置工具可供选择，这个领域的一些关键产品如下：

Puppet (<http://puppet.reductivelabs.com/>)：Ruby 写成的配置管理工具，使用 C/S 架构，使用 declarative language 配置客户端。

Cfengine (<http://www.cfengine.org>)：最先发布的开源配置工具之一，1993 年发布，同样是 C/S 架构，通常应用于教育机构。

LCFG (<http://www.lcfg.org/>)：C/S 架构的配置管理工具，使用 XML 定义配置。

Bcfg2：Python 编写的 C/S 架构的配置管理工具，使用规格书和客户机响应配置目标主机。

我们工作中主要是用 Puppet 来自动化部署 Linux/Unix 服务器，下面我简单向大家介绍一下它的用法。

1. Puppet 的简单介绍

(1) Puppet 的用途

Puppet 是开源的基于 Ruby 的系统配置管理工具，依赖于 C/S 的部署架构。主要开发者是 Luke Kanies，遵循 GPLv2 版权协议。从 1997 年开始 Kanies 参与了 Unix 的系统管理工作，Puppet 的开发源于这些经验。因为对已有的配置工具不甚满意，在 2001 年到 2005 年期间，Kanies 开始在 Reductive 实验室里从事工具的开发。很快，Reductive 实验室发布了他们的旗舰产品——Puppet。

(2) Puppet 的特性

许多系统配置管理工作的方式非常类似，如 cfengine。那是什么让 Puppet 与众不同呢？

Puppet 的语法允许你创建一个单独脚本，用来在你所有的目标主机上建立一个用户。所有的目标主机会依次使用适合本地系统的语法来解释和执行这个模块。如果这个配置是在 RedHat 服务器上执行的，建立用户使用 useradd 命令；如果这个配置是在 FreeBSD 主机上执行的，使用的是 adduser 命令。

Puppet 另一个卓越的地方是它的灵活性。源于开源软件的天性，你可以自由地获得 Puppet 的源码，如果你遇到问题并且有能力处理的话，你可以修改或加强 Puppet 的代码使其适用于你的环境，然后解决这个问题。另外，社区开发者和捐献者还在不断地增强 Puppet 的功能。一个大的开发者与用户社区也致力于提供 Puppet 的文档和技术支持。

Puppet 也是易于扩展的。定制软件包的支持功能和特殊的系统环境配置能够快速简单地添加至 Puppet 的安装程序中。

(3) Puppet 的工作模式

Puppet 是一个 C/S 架构的配置管理工具。在中央服务器上安装 puppet-server 软件包（被称做 Puppet

master), 在需要管理的目标主机上安装 Puppet 客户端软件 (被称做 Puppet Client)。当客户端连接上 Puppet master 后, 定义在 Puppet master 上的配置文件会被编译, 然后在客户端上运行。每个客户端默认每半个小时和服务器进行一次通信, 确认配置信息的更新情况。如果有新的配置信息或配置信息已经改变, 配置将会被重新编译并发布到各客户端执行。也可以在服务器上主动触发一个配置信息的更新, 强制各客户端进行配置。如果客户端的配置信息被改变, 它可以从服务器上获得原始配置进行校正。

(4) Puppet 的未来

Puppet 是一个年轻的工具, 仍然处于开发和发展中。Puppet 社区快速壮大起来, 并且有许多新的想法不断融入, 促使开发、更新和模块每天都在呈现, 现在多用于规模较大的分布式环境。

2. 安装 Puppet 前的准备工作

我们可以先准备两台 Centos5.5 x86_64 的机器, 做好安装前的准备工作, 两台机器都先用 ntpdate 对好时, 以防止实验时出现时间报错, 如下所示:

```
ntpdate ntp.api.bz
```

两台机器的主机名及 IP 分配如下:

server. cn7788. com 192. 168. 1. 5, Puppet 服务器端

slave. cn7788. com 192. 168. 1. 6, Puppet 客户端

我们也要配置好它们的主机名, 分别在两台机器的/etc/hosts 里添加如下内容:

```
192.168.1.5 server.cn7788.com
```

```
192.168.1.6 slave.cn7788.com
```

3. Puppet 的安装步骤 (这些步骤在两台机器上都要执行)

1) 安装企业 Linux (EPEL) yum 仓库的额外的软件包, 请大家注意 32 位与 64 位的区别, 我这里的系统是 64 位的, 所以我用的是适用于 64 位系统的 rpm 软件包, 命令如下:

```
rpm -Uvh http://download.fedora.redhat.com/pub/epel/5/x86_64/epel-release-5-4.noarch.rpm
```

2) 安装 Puppet 需要的软件包, 这里可以直接用 yum 安装。

```
yum install -y mysql mysql-devel mysql-server ruby ruby-devel ruby-irb ruby-mysql ruby-rdoc ruby-ri
```

我安装的版本是 ruby1.8.5, 不要安装 1.8.7 以上的版本, Puppet 还不支持。安装完 ruby 后, 我们看一下通过 yum 安装后的 yum 版本, 命令如下:

```
ruby --version
```

```
ruby 1.8.5 (2006-08-25) [x86_64-linux]
```

3) 下载最新版本的 Puppet, 尽量选择最新版的 Puppet, 它会修复之前版本的 Bug, 这也是我不推荐直接用 yum 安装的原因。

我用的是编译安装, 所以需要先安装 facter, 如下所示:

```
# wget http://puppetlabs.com/downloads/facter/facter-1.5.9.tar.gz
```

```
# tar zxvf facter-1.5.9
```

```
# cd facter-1.5.9
```

```
# /usr/bin/ruby install.rb
```

注意 facter 的作用是收集主机的一些资料，比如 CPU、主机 IP 等。facter 把收集的资料数据发送给 Puppet 服务器端，服务器端就可以根据不同的条件来对不同的节点机器生成不同的 Puppet 配置文件了。

4) 下载安装 Puppet 主程序，如下所示：

```
# wget http://puppetlabs.com/downloads/puppet/puppet-2.6.3.tar.gz
# tar zxvf puppet-2.6.3.tar.gz
# cd puppet-2.6.3
# /usr/bin/ruby install.rb
```

5) Server 端的配置步骤如下所示。

a) 拷贝源文件，配置 Puppet 的 Server 端，如下所示：

```
#mkdir /etc/puppet
#cp conf/auth.conf /etc/puppet/
#cp conf/redhat/filesserver.conf /etc/puppet/
#cp conf/redhat/puppet.conf /etc/puppet/
#cp conf/redhat/server.init /etc/init.d/puppetmaster
#chmod +x /etc/init.d/puppetmaster
#chkconfig - add puppetmaster
#chkconfig puppetmaster on
#mkdir -p /etc/puppet/manifests
```

b) 创建 Puppet 账号，如下所示：

```
puppetmasterd --mkusers
```

执行中可能会出现一些错误，基本上是以以前安装或创建过 Puppet 用户的原因，这些错误很好解决。我们执行下面这个命令就是让它自动去 `/var/lib/puppet` 下创建一些目录，如果遇到不能生成 Puppet 组和 Puppet 用户的情况，可以手动操作，命令如下：

```
groupadd puppet
useradd -g puppet puppet
```

c) 建立相应的目录，如下所示：

```
mkdir /var/lib/puppet/rrd
chown puppet.puppet /var/lib/puppet/rrd
```

d) 启动服务，如下所示：

```
/etc/init.d/puppetmaster start
```

第一次启动时 Puppet 会自动创建所需的文件，包括一系列证书文件等。

6) Slave 端的配置过程如下所示。

编译安装的过程与 Server 端相同。

a) 复制配置文件，配置 Slave 端的 Puppet，如下所示：

```
# mkdir /etc/puppet
# cp conf/auth.conf /etc/puppet/
# cp conf/namespaceauth.conf /etc/puppet/
```

```
# cp conf/redhat/puppet.conf /etc/puppet/
# cp conf/redhat/client.init /etc/init.d/puppet
# chmod +x /etc/init.d/puppet
# chkconfig --add puppet
# chkconfig puppet on
```

b) 创建 Puppet 账号，如下所示：

```
puppetd --mkusers
```

如果出现错误，可手动建立 Puppet 用户组及 Puppet 用户。

c) 建立 Puppet 相应的目录，如下所示：

```
mkdir -p /var/lib/puppet/rrd
chown puppet.puppet /var/lib/puppet/rrd
```

d) 启动服务，如下所示：

```
/etc/init.d/puppet start
```

注意 当启动 Puppet 时，程序会根据 puppet.conf 中的配置自动向服务端发起证书验证请求。

e) 然后向 Server 端发出请求命令，如下所示：

```
puppetd --test --server server.cn7788.com
warning: peer certificate won't be verified in this SSL session
info: Caching certificate for ca
warning: peer certificate won't be verified in this SSL session
warning: peer certificate won't be verified in this SSL session
info: Creating a new SSL certificate request for slave.cn7788.com
info: Certificate Request fingerprint (md5): D2:01:A0:C4:3A:C1:13:D2:56:75:BF:8D:82:78:FE:61
warning: peer certificate won't be verified in this SSL session
warning: peer certificate won't be verified in this SSL session
warning: peer certificate won't be verified in this SSL session
Exiting; no certificate found and waitforcert is disabled
```

f) Puppet 的 Server 端要用如下命令接受请求：

```
puppetca -s -a
notice: Signed certificate request for slave.cn7788.com
notice: Removing file Puppet::SSL::CertificateRequest slave.cn7788.com at
'/var/lib/puppet/ssl/ca/requests/slave.cn7788.com.pem'
```

此命令表示接受全部认证，我们也可以用 puppetca --list 查看所有需要认证的客户端。

g) Slave 端再发一次认证请求，如下所示：

```
puppetd --test --server server.cn7788.com
warning: peer certificate won't be verified in this SSL session
info: Caching certificate for slave.cn7788.com
info: Caching certificate_revocation_list for ca
info: Caching catalog for slave.cn7788.com
info: Applying configuration version '1308468459'
info: Creating state file /var/lib/puppet/state/state.yaml
```

```
notice: Finished catalog run in 0.02 seconds
```

出现如上字样时表示 Server 端已接受 Slave 端的请求，连接成功了。

4. Puppet 简单的文件应用

我们在服务端的/etc/puppet/manifests/下建立文件 site.pp，此文件可以将/tmp/andrew.txt 的内容和权限都推送过去。如果客户端存在着此文件，则用此文件定义的文件内容和权限；如果不存在，则推送过去，文件内容和权限由此文件定义。文件的内容如下：

```
node default{
  file {"/tmp/andrewy.txt":
    content => "hello, My Name is Andrew.Yu !\n",
    ensure => present,
    mode => 644,
    owner => root,
    group => root,
  }
}
```

记得两台机器都要关闭 iptables 和 SELinux。

关于文件内容，其他的都很好理解，只是 ensure => present 表示什么意思呢？

ensure 后面可以接许多参数，如果文件本来不存在会确认是否要新建文件，可以设置的值为：absent 和 present、file 及 directory。如果后面接的是 present，就会检查该文件是否存在，如果不存在就新建该文件。

客户机 slave.cn7788.com 执行如下命令，马上可以看到生成的文件，如下所示：

```
puppetd --test --server server.cn7788.com
info: Caching catalog for slave.cn7788.com
info: Applying configuration version '1308478614'
--- /tmp/andrewy.txt    2011-06-19 18:16:01.000000000 +0800
+++ /tmp/puppet - file.17509.0    2011-06-19 18:16:54.000000000 +0800
@@ -1 +1 @@
- hello My Name is Andrew.Yu!
+ hello, My Name is Andrew.Yu !
info: FileBucket adding /tmp/andrewy.txt as {md5}7cf138f4b5a29f15e5e6a99738c059c5
info: /Stage[main]//Node[default]/File[/tmp/andrewy.txt]: Filebucketed /tmp/andrewy.txt to
puppet with sum 7cf138f4b5a29f15e5e6a99738c059c5
notice: /Stage[main]//Node[default]/File[/tmp/andrewy.txt]/content: content changed '{md5}
7cf138f4b5a29f15e5e6a99738c059c5' to '{md5}b2ad2c2aaa2ad6776e527a7dff69d87d'
notice: Finished catalog run in 0.08 seconds
```

由于之前已经生成了文件，这里只是在原有文件的基础上发生了改变，Puppet 会清楚地显示前后文件内容的对比。以后服务器端的文件内容或权限发生更改时，直接在客户机上执行以上命令即可。我们可以在客户端上将此文件删除，然后再重新连接，你会发现服务器端又将此文件推送过来了，我们接着观察此文件的权限及属主属组等，如下所示：

```
ls -lsart andrewy.txt
4 -rw-r--r-- 1 root root 30 Jun 19 22:24 andrewy.txt
```


5. Puppet 的相关配置文件及路径

先来看看 Puppet 配置文件，其主配置文件（puppet.conf）如下所示：

(1) 配置文件的命名空间

- ☐ main：通用配置选项。
- ☐ puppetd：客户端配置选项。
- ☐ puppetmasterd：服务端配置选项。

(2) main 命名空间的选项

- ☐ confdir：配置文件目录，默认在/etc/puppet。
- ☐ vardir：动态数据目录，默认在/var/lib/puppet。
- ☐ logdir：日志目录，默认在/var/log/log。
- ☐ rundir puppet：PID 目录，默认在/var/run/puppet。
- ☐ statedir：state 目录，默认在\$vardir/state。
- ☐ statefile：state 文件，默认在\$statedir/state.yaml。
- ☐ sslidir：SSL 证书目录，默认在\$vardir/ssl。
- ☐ trace：发生错误时显示跟踪信息，默认为 false。
- ☐ filetimeout：检测配置文件状态改变的时间周期，单位为秒，默认是 15 秒。
- ☐ syslogfacility：指定 syslog 功能为 user 级，默认为 daemon 级。

(3) puppetmasterd 命名空间选项

- ☐ user：后台进程执行的用户。
- ☐ group：后台进程执行的组。
- ☐ manifestdir：manifests 文件存储目录，默认为\$confdir/manifests。
- ☐ manifest：manifest 站点文件的名字，默认为 site.pp。
- ☐ bindaddress：后台进程绑定的网卡地址接口。
- ☐ masterport：后台进程执行的端口，默认为 8140。

(4) puppet 命名空间选项

- ☐ server puppet：puppet 服务器名，默认为 puppet。
- ☐ runinterval seconds：puppet 应用配置的时间间隔，默认 1800 秒（0.5 小时）。
- ☐ puppetdlockfile file：puppet lock 文件位置，默认\$statedir/puppetdlock。
- ☐ puppetport port：后台进程执行的端口，默认 8139。
- ☐ 文件服务配置文件（fileserver.conf）：限制访问的客户端，有点类似于 NFS。

Puppet 适用的场景是 500 台以上规模的 Linux/Unix 分布式环境，它的优点可以在此环境中发挥得淋漓尽致。这个软件越来越成熟和强大了，它有着很好的发展前景。我目前的工作环境并非大规模的 CDN 节点，仅用 Xmanager3.0 + SHELL 脚本及 rsync + Inotify 就可以实现自动化工作了。所以这里只简单介绍了 Puppet 简单的安装和部署情况，真正的 Puppet 使用起来博大精深，有兴趣的朋友可以在工作中尝试研究 Puppet 更为强大的用法。

参考文档：

http://hi.baidu.com/nt_fs/blog/item/617e58dfc59cc20b632798bf.html

http://os.51cto.com/art/201011/232845_2.htm

<http://puppet.wikidot.com/> (Puppet 中文 Wiki)

附录 D 漫谈 CDN 系统运维与电子商务运维

说明：本附录是根据国内领先的 IT 技术网站 51CTO (www.51cto.com) 对本书作者的专访整理而成的，希望这篇文章对从事 CDN 系统运维和电子商务系统运维的朋友有所启发和帮助。

51CTO

你现在加盟“一拍网”，主要是负责电子商务网站的维护、内网开发环境的部署和技术研发吗？

抚琴煮酒

主要负责电子商务网站的维护和技术研发。由于公司目前规模较小，所以没有内网开发环境部署这一块，程序相关的工作我们全部外包出去了，公司没有程序员。“一拍网”的网站架构是我在负责设计，一开始的预算不多，因为前期规模不会很大，后期会再增加 4 台 Web 服务器和 4 台 squid 或 varnish 反向加速服务器。

51CTO

也就是说，你现在的工作更偏向于系统运维，从你的工作经历来看，你对 Windows 服务器、Linux/BSD 服务器、网络，以及数据库等都有一定的研究和认识。学了这么多技术，从 2005 年正式开始转型企业网管到现在，在你的职业发展过程中，有没有走过弯路？有经验或教训与大家分享一下吗？

抚琴煮酒

从我的经历来看，有两个方面我花了不少的时间和精力，但收获非常少。第一，有段时间我很喜欢研究别人没研究过的东西，比如 DDNS，花了不少时间和精力做出来的方案却无法在生产环境中应用。在这里我想提醒大家，学习的方向应该尽量与生产相结合，在实际的工作中，企业真正看重的是你的经验，即你所掌握的能够迅速投入生产的技术，而不是你会干什么。第二，我花了近一年时间从事 RHCE 和 Linux 方面的教学工作，虽然认识了不少朋友，也拓宽了自己的职业发展道路，但这项工作与我目前从事的工作还是有较大差别的。当讲师虽然能使自己的知识沉淀下来，但由于很难接触到生产环境，技术退步的可能性反而更大。

51CTO

在你看来，如果要在项目实施工程师方向有所发展，最有效的方式还是尽可能地多参与实际的项目，这样才能知道企业真正需要什么技术，是这样吗？

抚琴煮酒

没错！如果对自己要求比较高，也不怕辛苦，建议多参加生产环境的项目实践。另外，如果条件允许，建议到大公司和门户网站工作。因为，如果没有足够大的并发需求，许多开源的新技术都用不上。大家可以关注一下淘宝的新架构，它用了不少自己开发的新技术，这些应该是我们重点关注的。企业真正需要的并不是什么技术，而是要能做事的人，特别是能够为公司节约钱的人，这也是 XEN 虚拟化和 Linux 集群如此流行的原因之一。

51CTO

刚好你提到“节约钱”，你有没有遇到过老板出于成本考虑不愿意添加服务器的情况？

抚琴煮酒

这种情况很常见。我为很多小网站提供过服务，因为客户不愿意增加服务器，所以我被逼用 HAProxy 做 1+2 的架构。公司用的测试服务器就不说了，经常是组装的服务器，连 VMware ESXI 都装不上，所以我花了很多时间来测试 VMware Server 和思杰的 XenServer5.6。

51CTO

在你看来，维护 CDN 系统都需要掌握哪些技术？

抚琴煮酒

CDN 主要是用来解决高并发和南北互连问题的，南北互连的问题即电信和网通之间的问题。CDN 系统维护的技术含量非常高，我认为至少应该具备以下知识：

(1) 非常熟悉 Linux/FreeBSD 系统，经验至少在 3 年以上，因为很多 CDN 系统都是基于 FreeBSD 的。

(2) 非常熟悉 Linux 集群相关的技术，比如 LVS、Nginx、HAProxy、F5 等硬件负载均衡，以及软件级别的负载均衡，即 LVS/Nginx 和 HAProxy，最好能达到比较精的水平。

(3) 了解 squid 和 varnish 的原理，能够熟练地配置和优化它们，尤其是反响代理的优化。这是基本功，这也是使用 CDN 时最重要的技能之一。

(4) bind 的智能 view 的搭建和维护，这一点不用多说了，是基本功。

(5) 至少熟悉一种数据库（我个人的选择是 MySQL），能够与 DBA 一起交流，能够设计数据库架构和对数据库进行切分，能解决流量过大带来的数据库压力问题。

(6) 熟悉 iptables 的配置。

(7) 能够熟练地配置 OpenVPN，解决节点之间互连的问题。

(8) 能够用 SHELL 和 puppet 等分布式工具解决节点服务器自动配置与同步的问题。

CDN 维护的技术含量非常高，我这里的列出的知识点只少不多，有些技术问题可能暂时没想到。现在的 CDN 基本都是上亿级的 PV，并发一般都上万，这样的环境是最锻炼人的！

51CTO

CDN 系统的维护听起来很复杂，相比之下，电子商务网站的运维又有哪些需要掌握的技术呢？

抚琴煮酒

我认为维护电子商务网站应该掌握的技术有：

- (1) 熟悉安全技术，了解硬件防火墙的性能，能熟练地配置 iptables 和与 Linux/Unix 相关的安全工具。
- (2) 熟悉 Linux 集群相关的技术，比如 LVS、Nginx、HAProxy，了解其原理和会话保持机制。
- (3) 能熟练地配置 Nginx 和 Apache 服务器。
- (4) 至少熟悉一种数据库，比如 MySQL 或 Oracle。
- (5) 熟悉存储技术、虚拟化技术和 SHELL 脚本。
- (6) 有一定的 Java 或 PHP 编程基础。

51CTO

前一阵子你在研究“一拍网”到底是用 LVS + Keepalived 架构，还是 HAProxy + Heartbeat 架构，最终还是选择了 LVS + Keepalived 架构，这是出于哪方面考虑的呢？

抚琴煮酒

LVS + Keepalived 架构我在实际的生产环境中使用过多次，大家应该也比较熟悉，很多网站都是用的这种成熟的负载均衡高可用方案。HAProxy + Heartbeat 架构我目前只在内网测试环境下通过了，担心存在 Heartbeat 的问题，所以最后还是采用的 LVS + Keepalived。

HAProxy 和 LVS 很类似，主要是 Keepalived 和 Heartbeat 不一样。Keepalived 是主 Master 机通过发送 VRRP 包来模拟一个路由器，在这个虚拟路由器中，只有作为 Master 的 VRRP 路由器会一直发送 VRRP 广告包（VRRP Advertisement message），BACKUP 不会抢占 Master，这个虚拟的路由器 IP 即我们网站的 IP，而 Heartbeat 的工作机制大家应该都有所了解，它也是比较成熟的双机方案。HAProxy + Heartbeat 方案只是我感兴趣的研究方向，如果大家想用成熟的线上方案，我推荐 LVS + Keepalived 或 Nginx + Keepalived，相关内容请参考我的负载均衡的相关专题。

[G e n e r a l I n f o r m a t i o n]

书名 = 构建高可用 L i n u x 服务器

作者 = 余洪春著

页数 = 5 7 4

出版社 = 北京市：机械工业出版社

出版日期 = 2 0 1 2 . 0 1

S S 号 = 1 2 9 3 8 7 8 5

D X 号 = 0 0 0 0 0 8 1 9 2 1 1 6

U R L = h t t p : / / b o o k . s z d n e t . o r g . c n / b o o k D e t a i l . j s

p ? d x N u m b e r = 0 0 0 0 0 8 1 9 2 1 1 6 & d = 0 D F F D B 6 0 1 5 1 5 5 D E 9 E

0 A 5 B 3 7 3 7 5 B 1 F 5 F 7